



デベロッパーガイド

Amazon Elastic Compute Cloud



Amazon Elastic Compute Cloud: デベロッパーガイド

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

Amazon の商標およびトレードドレスは、Amazon のものではない製品またはサービスにも関連して、お客様に混乱を招いたり Amazon の信用を傷つけたり失わせたりするいかなる形においても使用することはできません。Amazon が所有していないその他のすべての商標は Amazon との提携、関連、支援関係の有無にかかわらず、それら該当する所有者の資産です。

Table of Contents

Amazon EC2 へのプログラムでのアクセス	1
サービスエンドポイント	2
IPv4 エンドポイント	2
デュアルスタック (IPv4 および IPv6) エンドポイント	2
リージョン別のサービスエンドポイント	3
エンドポイントの指定	9
結果整合性	11
べき等性	12
Amazon EC2 のべき等性	13
RunInstances のべき等性	16
例	17
べき等リクエストの再試行の推奨事項	19
API リクエストのスロットリング	20
スロットリングの適用方法	21
リクエストトークンの上限	22
リソーストークンの上限	32
API スロットリングをモニタリングする	33
再試行とエクスポネンシャルバックオフ	33
制限の引き上げのリクエスト	34
Pagination (ページ分割)	35
ベストプラクティス	36
一般的な問題	37
の使用AWS CLI	39
AWS CLI の詳細	39
CloudFormation の使用	40
Amazon EC2 と CloudFormation テンプレート	40
Amazon EC2 のリソース	40
CloudFormation の詳細はこちら	44
AWS SDK を使用する	45
Amazon EC2 API のコードサンプル	45
AWS SDK の詳細	45
Amazon EC2 の低レベル API	46
Console-to-Code	47
コードの例	48

基本	55
Hello Amazon EC2	61
基本を学ぶ	79
アクション	323
シナリオ	1236
レジリエントなサービスの構築と管理	1237
プライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成する	1405
Amazon VPC の使用を開始する	1423
Transit Gateway を開始する	1438
VPC IPAM の開始方法	1447
CloudWatch を使用して API リクエストをモニタリングする	1458
Amazon EC2 API メトリクスを有効にする	1458
Amazon EC2 API メトリクスおよびディメンション	1459
メトリクス	1459
ディメンション	1461
メトリクスデータ保持	1461
ユーザーに代わって行われるリクエストのモニタリング	1461
料金	1461
Amazon CloudWatch との連携	1461
CloudWatch メトリクスの表示	1462
CloudWatch アラームの作成	1463
.....	mcdlxv

Amazon EC2 へのプログラムでのアクセス

AWS マネジメントコンソール または プログラムインターフェイスを使用して、Amazon EC2 リソースを作成および管理できます。Amazon EC2 コンソールの使用方法については、「[Amazon EC2 ユーザーガイド](#)」を参照してください。

仕組み

- [Amazon EC2 エンドポイント](#)
- [結果整合性](#)
- [べき等性](#)
- [リクエストのロットリング](#)
- [Pagination \(ページ分割\)](#)

プログラミングインターフェイス

- [AWS Command Line Interface \(AWS CLI\)](#)
- [AWS CloudFormation](#)
- [AWS SDK](#)
- [低レベル API](#)

はじめに

- [コードの例](#)
- [Console-to-Code](#)

モニタリング

- [AWS CloudTrail](#)
- [リクエストのモニタリング](#)

Amazon EC2 サービスエンドポイント

エンドポイントは、AWS ウェブサービスのエン트리ポイントとして機能する URL です。Amazon EC2 は、次のエンドポイントタイプをサポートします。

- [IPv4 エンドポイント](#)
- [デュアルスタックエンドポイント](#) (IPv4 と IPv6 の両方をサポート)
- [FIPS エンドポイント](#)

リクエストを行うと、使用するエンドポイントを指定できます。エンドポイントを指定しない場合、デフォルトで IPv4 エンドポイントが使用されます。別のエンドポイントタイプを使用するには、リクエストで指定する必要があります。これを行う方法の例については、「[エンドポイントの指定](#)」を参照してください。利用可能なエンドポイントの表については、「[リージョン別のサービスエンドポイント](#)」を参照してください。

IPv4 エンドポイント

IPv4 エンドポイントは IPv4 トラフィックのみをサポートします。IPv4 エンドポイントは、すべてのリージョンで利用できます。

一般的なエンドポイントである `ec2.amazonaws.com` を指定する場合は、`us-east-1` のエンドポイントを使用します。別のリージョンを使用するには、関連するエンドポイントを指定します。例えば、`ec2.us-east-2.amazonaws.com` をエンドポイントとして指定した場合、リクエストは `us-east-2` エンドポイントに転送されます。

IPv4 エンドポイント名では、次の命名規則が使用されます。

- `service.region.amazonaws.com`

例えば、`eu-west-1` リージョンの IPv4 エンドポイントは、`ec2.eu-west-1.amazonaws.com` です。

デュアルスタック (IPv4 および IPv6) エンドポイント

デュアルスタックエンドポイントは、IPv4 と IPv6 トラフィックの両方をサポートします。デュアルスタックエンドポイントにリクエストを行うと、エンドポイント URL は、ネットワークとクライアントが使用するプロトコルに応じて IPv6 または IPv4 アドレスに解決されます。

Amazon EC2 はリージョンのデュアルスタックエンドポイントのみをサポートしているため、エンドポイント名の一部としてリージョンを指定する必要があります。デュアルスタックエンドポイント名には、次の命名規則が使用されます。

- `ec2.region.api.aws`

例えば、eu-west-1 リージョンのデュアルスタックエンドポイント名は、`ec2.eu-west-1.api.aws` です。

リージョン別のサービスエンドポイント

Amazon EC2 のサービスエンドポイントは次のとおりです。リージョンの詳細については、「Amazon EC2 ユーザーガイド」の「[リージョンとアベイラビリティゾーン](#)」を参照してください。

リージョン名	リージョン	エンドポイント	プロトコル
米国東部 (オハイオ)	us-east-2	ec2.us-east-2.amazonaws.com	HTTP
		ec2-fips.us-east-2.amazonaws.com	および HTTPS
		ec2.us-east-2.api.aws	HTTPS
			HTTPS
米国東部 (バージニア北部)	us-east-1	ec2.us-east-1.amazonaws.com	HTTP
		ec2-fips.us-east-1.amazonaws.com	および HTTPS
		ec2.us-east-1.api.aws	HTTPS
			HTTPS
米国西部 (北カリフォルニア)	us-west-1	ec2.us-west-1.amazonaws.com	HTTP
		ec2-fips.us-west-1.amazonaws.com	および HTTPS
		ec2.us-west-1.api.aws	HTTPS
			HTTPS

リージョン名	リージョン	エンドポイント	プロトコル	
			HTTPS	
米国西部 (オレゴン)	us-west-2	ec2.us-west-2.amazonaws.com	HTTP	
		ec2-fips.us-west-2.amazonaws.com	および HTTPS	
		ec2.us-west-2.api.aws	HTTPS	
			HTTPS	
アフリカ (ケープタウン)	af-south-1	ec2.af-south-1.amazonaws.com	HTTP	
		ec2.af-south-1.api.aws	および HTTPS	
			HTTPS	
アジアパシフィック (香港)	ap-east-1	ec2.ap-east-1.amazonaws.com	HTTP	
		ec2.ap-east-1.api.aws	および HTTPS	
			HTTPS	
アジアパシフィック (ハイデラバード)	ap-south-2	ec2.ap-south-2.amazonaws.com	HTTPS	
アジアパシフィック (ジャカルタ)	ap-southeast-3	ec2.ap-southeast-3.amazonaws.com	HTTPS	
アジアパシフィック (マレーシア)	ap-southeast-5	ec2.ap-southeast-5.amazonaws.com	HTTPS	

リージョン名	リージョン	エンドポイント	プロトコル	
アジアパシフィック (メルボルン)	ap-southeast-4	ec2.ap-southeast-4.amazonaws.com	HTTPS	
アジアパシフィック (ムンバイ)	ap-south-1	ec2.ap-south-1.amazonaws.com ec2.ap-south-1.api.aws	HTTP および HTTPS HTTPS	
アジアパシフィック (ニュージーランド)	ap-southeast-6	ec2.ap-southeast-6.amazonaws.com	HTTPS	
アジアパシフィック (大阪)	ap-northeast-3	ec2.ap-northeast-3.amazonaws.com	HTTP および HTTPS	
アジアパシフィック (ソウル)	ap-northeast-2	ec2.ap-northeast-2.amazonaws.com ec2.ap-northeast-2.api.aws	HTTP および HTTPS HTTPS	
アジアパシフィック (シンガポール)	ap-southeast-1	ec2.ap-southeast-1.amazonaws.com ec2.ap-southeast-1.api.aws	HTTP および HTTPS HTTPS	

リージョン名	リージョン	エンドポイント	プロトコル	
アジアパシフィック (シドニー)	ap-southeast-2	ec2.ap-southeast-2.amazonaws.com ec2.ap-southeast-2.api.aws	HTTP および HTTPS HTTPS	
アジアパシフィック (台北)	ap-east-2	ec2.ap-east-2.amazonaws.com	HTTPS	
アジアパシフィック (タイ)	ap-southeast-7	ec2.ap-southeast-7.amazonaws.com	HTTPS	
アジアパシフィック (東京)	ap-northeast-1	ec2.ap-northeast-1.amazonaws.com ec2.ap-northeast-1.api.aws	HTTP および HTTPS HTTPS	
カナダ (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 および HTTPS HTTPS HTTPS	
カナダ西部 (カルガリー)	ca-west-1	ec2.ca-west-1.amazonaws.com ec2-fips.ca-west-1.amazonaws.com	HTTPS HTTPS	
欧州 (フランクフルト)	eu-central-1	ec2.eu-central-1.amazonaws.com ec2.eu-central-1.api.aws	HTTP および HTTPS HTTPS	

リージョン名	リージョン	エンドポイント	プロトコル	
欧州 (アイルランド)	eu-west-1	ec2.eu-west-1.amazonaws.com ec2.eu-west-1.api.aws	HTTP および HTTPS HTTPS	
欧州 (ロンドン)	eu-west-2	ec2.eu-west-2.amazonaws.com ec2.eu-west-2.api.aws	HTTP および HTTPS HTTPS	
ヨーロッパ (ミラノ)	eu-south-1	ec2.eu-south-1.amazonaws.com ec2.eu-south-1.api.aws	HTTP および HTTPS HTTPS	
欧州 (パリ)	eu-west-3	ec2.eu-west-3.amazonaws.com ec2.eu-west-3.api.aws	HTTP および HTTPS HTTPS	
欧州 (スペイン)	eu-south-2	ec2.eu-south-2.amazonaws.com	HTTPS	
欧州 (ストックホルム)	eu-north-1	ec2.eu-north-1.amazonaws.com ec2.eu-north-1.api.aws	HTTP および HTTPS HTTPS	
欧州 (チューリッヒ)	eu-central-2	ec2.eu-central-2.amazonaws.com	HTTPS	

リージョン名	リージョン	エンドポイント	プロトコル	
イスラエル (テルアビブ)	il-central-1	ec2.il-central-1.amazonaws.com	HTTPS	
メキシコ (中部)	mx-central-1	ec2.mx-central-1.amazonaws.com	HTTPS	
中東 (バーレーン)	me-south-1	ec2.me-south-1.amazonaws.com ec2.me-south-1.api.aws	HTTP および HTTPS HTTPS	
中東 (アラブ首長国連邦)	me-central-1	ec2.me-central-1.amazonaws.com	HTTPS	
南米 (サンパウロ)	sa-east-1	ec2.sa-east-1.amazonaws.com ec2.sa-east-1.api.aws	HTTP および HTTPS HTTPS	
AWS GovCloud (米国東部)	us-gov-east-1	ec2.us-gov-east-1.amazonaws.com ec2.us-gov-east-1.api.aws	HTTPS HTTPS	
AWS GovCloud (米国西部)	us-gov-west-1	ec2.us-gov-west-1.amazonaws.com ec2.us-gov-west-1.api.aws	HTTPS HTTPS	

エンドポイントの指定

このセクションでは、リクエストを行うときにエンドポイントを指定する方法を例で示します。

AWS CLI

次の例は、AWS CLIを使用して us-east-2 リージョンのエンドポイントを指定する方法を示しています。

- デュアルスタック

```
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

次の例は、AWS SDK for Java 2.xを使用して us-east-2 リージョンのエンドポイントを指定する方法を示しています。

- デュアルスタック

```
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

次の例は、AWS SDK for Java 1.x を使用して eu-west-1 リージョンのエンドポイントを指定する方法を示しています。

- デュアルスタック

```
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

次の例は、AWS SDK for Go を使用して us-east-1 リージョンのエンドポイントを指定する方法を示しています。

- デュアルスタック

```
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")
})
```

Amazon EC2 API の結果整合性

Amazon EC2 API は、この API をサポートしているシステムの分散特性により、結果整合性モデルに従います。つまり、Amazon EC2 リソースに影響を及ぼす、実行した API コマンドの結果は、その後に実行するすべてのコマンドにすぐには表示されない可能性があります。前の API コマンドの直後に API コマンドを実行する場合は、この点に留意する必要があります。

結果整合性は、リソースの管理方法に影響を与える可能性があります。例えば、コマンドを実行してリソースを作成すると、最終的に他のコマンドに表示されます。つまり、その作成したリソースを変更または記述するコマンドを実行すると、その ID がシステム全体に伝播されず、リソースが存在しないことを示すエラーが表示されます。

結果整合性を管理するには、以下を実行します。

- コマンドを実行して変更する前に、リソースの状態を確認します。エクスポネンシャルバックオフアルゴリズムを使用して適切な Describe コマンドを実行し、前のコマンドがシステム内を伝播するのに十分な時間を確保できるようにします。これを行うには、数秒の待機時間から始めて Describe コマンドを繰り返し実行し、数分間の待機時間まで徐々に増やします。
- Describe コマンドが正確なレスポンスを返した場合でも、後続のコマンド間の待機時間を追加します。数秒の待機時間から始まる指数バックオフアルゴリズムを適用し、数分間まで徐々に増やします。

結果整合性エラーの例

以下は、結果整合性の結果として発生する可能性のあるエラーコードの例です。

- InvalidInstanceID.NotFound

RunInstances コマンドを正常に実行し、RunInstances のレスポンスで指定されたインスタンス ID を使用して別のコマンドをすぐに実行すると、InvalidInstanceID.NotFound エラーが返される可能性があります。これは、インスタンスが存在しないことを意味するものではありません。

影響を受ける可能性のある特定のコマンドは次のとおりです。

- DescribeInstances: インスタンスの実際の状態を確認するには、エクスポネンシャルバックオフアルゴリズムを使用してこのコマンドを実行します。
- TerminateInstances: インスタンスの状態を確認するには、まずエクスポネンシャルバックオフアルゴリズムを使用して DescribeInstances コマンドを実行します。

⚠ Important

`TerminateInstances` の実行後に `InvalidInstanceID.NotFound` エラーが発生しても、インスタンスが終了していたり、終了するわけではありません。インスタンスはまだ実行中である可能性があります。そのため、`DescribeInstances` を使用してインスタンスの状態を最初に確認することが重要です。

- `InvalidGroup.NotFound`

`CreateSecurityGroup` コマンドを正常に実行し、`CreateSecurityGroup` のレスポンスで提供されたセキュリティグループ ID を使用して別のコマンドをすぐに実行すると、`InvalidGroup.NotFound` エラーが返される場合があります。セキュリティグループの状態を確認するには、エクスポネンシャルバックオフアルゴリズムを使用して `DescribeSecurityGroups` コマンドを実行します。

- `InstanceLimitExceeded`

指定されたインスタンスタイプに対して現在のインスタンスの上限を超えるインスタンスをリクエストしました。インスタンスを起動してすぐに終了する場合に、この上限に予期せず到達する可能性があります。これは、インスタンスが終了してからしばらくの間、終了したインスタンスがインスタンスの上限に対してカウントされるためです。

Ensuring idempotency in Amazon EC2 API requests

変異する API リクエストを行うと、通常、リクエストはオペレーションの非同期ワークフローが完了する前に結果を返します。リクエストが既に結果を返している場合でも、操作が完了する前にタイムアウトしたり、その他のサーバーの問題が発生したりすることもあります。これにより、リクエストが成功したかどうかを判断するのが難しくなり、操作を正常に完了するために複数回の再試行が行われることがあります。ただし、元のリクエストとその後の再試行が成功すると、操作は複数回完了します。つまり、意図したよりも多くのリソースが作成される可能性があります。

冪等性とは、API リクエストが 1 回だけ完了することを保証するものです。冪等性リクエストでは、元のリクエストが正常に完了した場合、その後の再試行は追加のアクションを実行せずに正しく完了します。ただし、結果には、現在の作成ステータスなどの更新された情報が含まれる場合があります。

内容

- [Amazon EC2 のべき等性](#)

- [RunInstances のべき等性](#)
- [例](#)
- [べき等リクエストの再試行の推奨事項](#)

Amazon EC2 のべき等性

以下の API アクションはデフォルトでべき等であり、追加の設定は必要ありません。対応する AWS CLI コマンドも、デフォルトでべき等性をサポートしています。

デフォルトでべき等

- AssociateAddress
- CreateVpnConnection
- DisassociateAddress
- ReplaceNetworkAclAssociation
- TerminateInstances

次の API アクションは、オプションでクライアントトークンを使用したべき等性をサポートしています。対応する AWS CLI コマンドも、クライアントトークンを使用したべき等性をサポートしています。クライアントトークンは、最大 64 バイトの ASCII 文字で、大文字と小文字を区別する一意の文字列です。これらのアクションのいずれかを使用してべき等 API リクエストを行うには、リクエストでクライアントトークンを指定します。同じクライアントトークンを他の API リクエストに再利用しないでください。同じクライアントトークンと同じパラメータを使用して正常に完了したリクエストを再試行すると、それ以上のアクションを実行せずに再試行が成功します。同じクライアントトークンを使用して成功したリクエストを再試行しても、リージョンまたはアベイラビリティゾーン以外の 1 つ以上のパラメータが異なる場合、再試行は `IdempotentParameterMismatch` エラーで失敗します。

クライアントトークンを使用したべき等性

- 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

べき等性のタイプ

- リージョン – リクエストは各リージョンでべき等です。ただし、同じクライアントトークンを含む同じリクエストを別のリージョンで使用できます。
- ゾーン – リクエストは、リージョン内の各アベイラビリティゾーンでべき等です。たとえば、同じリージョンの AllocateHosts への 2 回の呼び出しで同じクライアントトークンを指定した場合、AvailabilityZone パラメータに異なる値を指定すると、呼び出しは成功します。

RunInstances のべき等性

[RunInstances](#) API アクションは、リージョンとゾーンの両方のべき等性を使用します。

使用されるべき等性のタイプは、RunInstances API リクエストでアベイラビリティゾーンを指定する方法によって異なります。リクエストは、次の場合にゾーンのべき等性を使用します。

- プレイスメントデータ型で AvailabilityZone パラメータを使用してアベイラビリティゾーンを明示的に指定する場合
- SubnetId パラメータを使用して暗黙的にアベイラビリティゾーンを指定する場合

アベイラビリティゾーンを明示的または暗黙的に指定しない場合、リクエストではリージョンのべき等性を使用します。

ゾーンのべき等性

ゾーンのべき等性により、RunInstances API リクエストがリージョン内の各アベイラビリティゾーンでべき等であることが保証されます。これにより、同じクライアントトークンを使用するリクエストは、リージョン内の各アベイラビリティゾーン内で 1 回だけ完了できます。ただし、同じクライアントトークンを使用して、リージョン内の他のアベイラビリティゾーンでインスタンスを起動できます。

たとえば、べき等リクエストを送信して us-east-1a アベイラビリティゾーンでインスタンスを起動し、us-east-1b アベイラビリティゾーンのリクエストで同じクライアント

トークンを使用する場合、それらの各アベイラビリティーゾーンでインスタンスを起動します。1 つ以上のパラメータが異なる場合、それらのアベイラビリティーゾーンで同じクライアントトークンを使用した後続の再試行は、それ以上のアクションを実行せずに正常に返されるか、IdempotentParameterMismatch エラーで失敗します。

リージョンのべき等性

リージョンのべき等性により、RunInstances API リクエストがリージョンでべき等であることが保証されます。これにより、同じクライアントトークンを使用するリクエストは、リージョン内で 1 回のみ完了できます。ただし、同じクライアントトークンを使用するまったく同じリクエストを使用して、別のリージョンでインスタンスを起動できます。

たとえば、us-east-1 リージョンでインスタンスを起動するべき等リクエストを送信し、次に eu-west-1 リージョンのリクエストで同じクライアントトークンを使用する場合、それらの各リージョンでインスタンスを起動します。1 つ以上のパラメータが異なる場合、それらのリージョンで同じクライアントトークンを使用した後続の再試行は、それ以上のアクションを実行せずに正常に返されるか、IdempotentParameterMismatch エラーで失敗します。

Tip

リクエストされたリージョンのアベイラビリティーゾーンのいずれかが利用できない場合、リージョンのべき等性を使用する RunInstances リクエストは失敗する可能性があります。AWS インフラストラクチャが提供するアベイラビリティーゾーンの機能を活用するには、インスタンスの起動時にゾーンのべき等性を使用することをお勧めします。ゾーンのべき等性を使用し、利用可能なアベイラビリティーゾーンをターゲットとする RunInstances リクエストは、リクエストされたリージョン内の別のアベイラビリティーゾーンが利用できない場合でも成功します。

例

AWS CLI コマンドの例

AWS CLI コマンドをべき等にするには、`--client-token` オプションを追加します。

例 1: べき等性

次の [allocate-hosts](#) コマンドは、クライアントトークンを含むため、べき等性を使用します。

```
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
```

例 2: run-instances リージョンのべき等性

次の [run-instances](#) コマンドは、クライアントトークンを含むため、リージョンのべき等性を使用しますが、明示的にも暗黙的にもアベイラビリティゾーンを指定しません。

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

例 3: run-instances ゾーンのべき等性

次の [run-instances](#) コマンドは、クライアントトークンと明示的に指定されたアベイラビリティゾーンを含むため、ゾーンのべき等性を使用します。

```
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 リクエストの例

API リクエストをべき等にするには、ClientToken パラメータを追加します。

例 1: べき等性

次の [AllocateHosts](#) API リクエストは、クライアントトークンを含むためべき等性を使用します。

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

例 2: RunInstances リージョンのべき等性

次の [RunInstances](#) API リクエストは、クライアントトークンを含むため、リージョンのべき等性を使用しますが、明示的にも暗黙的にもアベイラビリティゾーンを指定しません。

```
https://ec2.amazonaws.com/?Action=RunInstances
```

```
&ImageId=ami-3ac33653
&MaxCount=1
&MinCount=1
&KeyName=my-key-pair
&ClientToken=550e8400-e29b-41d4-a716-446655440000
&AUTHPARAMS
```

例 3: RunInstances ゾーンのべき等性

次の [RunInstances](#) API リクエストでは、クライアントトークンと明示的に指定されたアベイラビリティゾーンが含まれているため、ゾーンのべき等性を使用します。

```
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
```

べき等リクエストの再試行の推奨事項

次の表は、べき等 API リクエストに対して返される一般的な応答と、再試行の推奨事項を示しています。

応答	推奨事項	コメント
200 (OK)	再試行しないでください	元のリクエストは正しく完了しています。それ以降に再試行しても正常に戻ります。
400 シリーズ応答コード (クライアントエラー)	再試行しないでください	次のうち、リクエストに問題があります。 - 無効なパラメータまたはパラメータの組み合わせが含まれています。 - 権限のないアクションまたはリソースを使用しています。

応答	推奨事項	コメント
		状態の変更処理中のリソースを使用しています。 リクエストに状態の変更処理中のリソースが含まれている場合、リクエストを再試行すると成功する可能性があります。
500 シリーズ応答コード (サーバーエラー)	再試行	このエラーは AWS サーバー側の問題によって発生し、通常は一時的なものです。適切なバックオフ戦略でリクエストを繰り返してください。

Amazon EC2 API リクエストのロットリング

Amazon EC2 は、リージョンごとに各 AWS アカウントの EC2 API リクエストを調整します。これは、サービスのパフォーマンスを向上し、Amazon EC2 のすべてのユーザーに平等にご利用いただくために行われるものです。ロットリングにより、Amazon EC2 API へのリクエストが API リクエストの最大許容上限を超えないことが保証されます。API リクエストは、送信元が以下であるかどうかにかかわらず、リクエストの上限の対象となります。

- サードパーティーのアプリケーション
- コマンドラインツール
- Amazon EC2 コンソール

API スロットリング上限を超えると、RequestLimitExceeded エラーコードが返されます。

内容

- [スロットリングの適用方法](#)
- [リクエストトークンバケットのサイズとリフィルレート](#)
- [リソーストークンバケットのサイズとリフィルレート](#)
- [API スロットリングをモニタリングする](#)

- [再試行とエクスポネンシャルバックオフ](#)
- [制限の引き上げのリクエスト](#)

スロットリングの適用方法

Amazon EC2 では [トークンバケットアルゴリズム](#) を使用して、API スロットリングを実装します。このアルゴリズムでは、アカウントには、特定の数のトークンを保持するバケットがあります。バケット内のトークンの数は、特定の秒におけるスロットリング上限を表します。

Amazon EC2 では 2 種類の API スロットリングを実装します。

API スロットリングタイプ

- [リクエストレート制限](#)
- [リソースレート制限](#)

リクエストレート制限

リクエストレート制限により各 API が個別に評価され、ユーザーは、API ごとに行うリクエストの数に応じてスロットリングされます。リクエストを行うことに、API のバケットからトークンが 1 つ削除されます。例えば、DescribeHosts (非変化 API アクション) のトークンバケットサイズは 100 トークンです。1 秒あたり最大 100 個の DescribeHosts リクエストを行うことができます。1 秒あたりのリクエストが 100 個を超えると、その API でスロットリングされ、その秒内の残りのリクエストは失敗しますが、他の API のリクエストは影響を受けません。

バケットは設定されたレートで自動的に補充されます。バケットが最大容量に達していない場合、最大容量に達するまで、設定した数のトークンが毎秒追加されます。リフィルトークンが到着したときにバケットが満杯である場合、リフィルトークンは破棄されます。バケットは最大数を超えてトークンを保持することはできません。例えば、DescribeHosts (非変化 API アクション) のバケットサイズが 100 トークンで、リフィルレートが 1 秒あたり 20 トークンであるとしします。1 秒間に 100 個の DescribeHosts リクエストを行うと、バケットはゼロ (0) トークンまで減らされます。バケットは、最大容量の 100 トークンに達するまで、毎秒最大 20 トークンで補充されます。つまり、空のバケットは、その間にリクエストが行われなかった場合、5 秒後に最大容量に達します。

API リクエストを行う前に、バケットが完全にいっぱいになるまで待つ必要はありません。リフィルトークンはバケットに追加されると使用できます。リフィルトークンをすぐに使用しても、バケットが最大容量に達することはありません。例えば、DescribeHosts (非変化 API アクション) のバケットサイズが 100 トークンで、リフィルレートが 1 秒あたり 20 トークンであるとしします。1 秒あ

たり 100 個の API リクエストを行ってバケットを使い果たした場合、バケットに追加されるリフィルトークンを使用して、1 秒あたり 20 個の API リクエストを引き続き行うことができます。バケットは、1 秒あたり 20 個未満の API リクエストを行う場合のみ、最大容量に補充できます。

詳細については、「[リクエストトークンバケットのサイズとリフィルレート](#)」を参照してください。

リソースレート制限

次の表で説明されるように、RunInstances や TerminateInstances などの一部の API アクションでは、リクエストレート制限に加えてリソースレート制限を使用します。これらの API アクションには、リクエストの影響を受けるリソースの数に基づいて消費される個別のリソーストークンバケットがあります。リクエストトークンバケットと同様に、リソーストークンバケットにはバーストできるバケットの最大数と、必要な期間にわたって安定したリクエストレートを維持できるリフィルレートがあります。バケットが次の API リクエストをサポートするために補充されていない場合など、API の特定のバケットの上限を超えた場合、API スロットル制限の合計に達していても、API のアクションは制限されます。

例えば、RunInstances のリソーストークンバケットサイズは 1000 トークンで、リフィルレートは 1 秒あたり 2 トークンです。したがって、1000 インスタンスの 1 個のリクエストや 250 インスタンスの 4 個のリクエストなど、任意の数の API リクエストを使用して、1000 インスタンスをすぐに起動できます。リソーストークンバケットが空になったら、2 つのインスタンスに対して 1 つのリクエストを使用するか、1 つのインスタンスに対して 2 つのリクエストを使用して、毎秒最大 2 つのインスタンスを起動できます。

詳細については、「[リソーストークンバケットのサイズとリフィルレート](#)」を参照してください。

リクエストトークンバケットのサイズとリフィルレート

リクエストレート制限の目的で、API アクションは次のカテゴリに分類されます。

- 非変化アクション — リソースに関するデータを取得する API アクション。このカテゴリには通常、DescribeRouteTables、SearchTransitGatewayRoutes、GetIpamPoolCidrs などの、すべての Describe*、List*、Search*、Get* API アクションが含まれます。これらの API アクションは通常、API スロットリング上限が最も高くなります。
- フィルタリングされていない非変化アクションとページ分割されていない非変化アクション — 非変化 API アクションの特定のサブセット。[\[ページ分割\]](#) または [\[フィルター\]](#) を指定せずにリクエストされた場合に、より小さなトークンバケットのトークンを使用します。トークンが標準 (より大きな) トークンバケットから引かれるように、ページ分割とフィルタリングを使用することをお勧めします。

- **変化アクション** — リソースを作成、変更、または削除する API アクション。このカテゴリには通常、AllocateHosts、ModifyHosts、CreateCapacityReservation などの 非変化アクションとして分類されていないすべての API アクションが含まれます。これらのアクションは、非変化 API アクションよりもスロットリング上限が低くなります。
- **リソース集約型アクション** — 完了するのに最も時間がかかり、最も多くのリソースを消費する変化 API アクション。これらのアクションのスロットリング上限は、変化アクションよりもさらに低くなります。これらは、他の変化アクションとは別にスロットリングされます。
- **コンソールの非変化アクション** — Amazon EC2 コンソールからリクエストされる非変化 API アクション。これらの API アクションは、他の非変化 API アクションとは別にスロットリングされます。
- **未分類アクション** — これらは、定義上、他のカテゴリのいずれかに該当していても、独自のトークンバケットサイズとリフィルレートを受け取る API アクションです。

API アクションカテゴリ	アクション	バケットの最大容量	バケットリフィルレート
非変化アクション	別のカテゴリに含まれていない Describe*、List*、Search*、API アクション。	100	20
フィルタリングされていない非変化アクションとページ分割されていない非変化アクション	- DescribeInstances - DescribeInstanceStatus - DescribeNetworkInterfaces	50	10

API アクションカテゴリ	アクション	バケットの最大容量	バケットリフィルレート
	DescribeSecurityGroups - DescribeSnapshots - DescribeSpotInstanceRequests - DescribeVolumes		
変化アクション	リソース集約型アクションでも未分類アクションでもないすべての変化 API アクション。	50	5

API アクションカテゴリ	アクション	バケットの最大容量	バケットリフィルレート
リソース集約型アクション	- AcceptVpcPeeringConnection - AuthorizeSecurityGroupIngress - CancelSpotInstanceRequests - CreateKeyPair - CreateVpcPeeringConnection - DeleteVpcPeeringConnection - RejectVpcPeeringConnection - RevokeSecurityGroupIngress - RequestSpotInstances	50	5

API アクションカテゴリ	アクション	バケットの最大容量	バケットリフィルレート
コンソール非変化アクション	Amazon EC2 コンソールによって呼び出されますが、別のカテゴリには含まれない Describe*、List*、Search*、API アクション。	100	10

未分類アクション	バケットの最大容量	バケットリフィルレート
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

未分類アクション	バケットの最大容量	バケットリフィルレート
CancelDeclarativePoliciesReport	1	1
CopyImage	100	1
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

未分類アクション	バケットの最大容量	バケットリフィルレート
CreateVerifiedAccessTrustProvider	10	2
CreateVolume	100	5
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

未分類アクション	バケットの最大容量	バケットリフィルレート
DeleteVpcEndpoints	4	0.3
DeleteVpcEndpointServiceConfigurations	10	1
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

未分類アクション	バケットの最大容量	バケットリフィルレート
DescribeSpotFleetInstances	100	5
DescribeSpotFleetRequestHistory	100	5
DescribeSpotFleetRequests	50	3
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

未分類アクション	バケットの最大容量	バケットリフィルレート
GetDeclarativePoliciesReportSummary	5	5
GetHostReservationPurchasePreview	5	2
ModifyImageAttribute	100	5
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

未分類アクション	バケットの最大容量	バケットリフィルレート
PurchaseCapacityBlock	10	0.15
PurchaseCapacityBlockExtension	10	0.15
PurchaseHostReservation	5	2
PurchaseReservedInstancesOffering	5	5
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

リソーストークンバケットのサイズとリフィルレート

次の表に、リソースレート制限を使用する API アクションのリソーストークンバケットのサイズとリフィルレートを示します。

API アクション	バケットの最大容量	バケットリフィルレート
RunInstances	1,000	2

API アクション	バケットの最大容量	バケットリフィルレート
TerminateInstances	1,000	20
StartInstances	1,000	2
StopInstances	1,000	20

API スロットリングをモニタリングする

Amazon CloudWatch を使用して Amazon EC2 API リクエストをモニタリングし、API スロットリングに関するメトリクスを収集および追跡できます。API スロットリング上限に近づいたときに警告するアラームを作成することもできます。詳細については、「[Amazon CloudWatch を使用して Amazon EC2 API リクエストをモニタリングする](#)」を参照してください。

再試行とエクスポネンシャルバックオフ

アプリケーションで、API リクエストの再試行が必要な場合があります。例えば、次のようになります。

- リソースのステータスの更新を確認するには
- 多数のリソース (すべてのボリュームなど) を列挙するには
- サーバーエラー (5xx) またはスロットリングエラーでリクエストが失敗した後に再試行するには

ただし、クライアントエラー (4xx) は、リクエスト再試行する前にリクエストを修正して問題を解決する必要があります。

リソースの状態変更

状態変更を確認するためのポーリングを開始する前に、リクエストの予想完了時間を指定します。例えば、インスタンスがアクティブかどうかを確認するまで数分待ちます。ポーリングを開始する時に、連続するリクエストの間に適切なスリープ間隔を使用して API リクエストレートを下げます。最良の結果を得るには、漸増または可変スリープ間隔を使用します。

または、Amazon EventBridge を使用して、一部のリソースのステータスを通知することもできます。例えば、EC2 インスタンスの状態変更通知イベントを使用して、インスタンスの状態変更を通知できます。詳細については、「[EventBridge を使用して Amazon EC2 を自動化する](#)」を参照してください。

再試行

API リクエストをポーリングまたは再試行する必要がある場合は、エクスポネンシャルバックオフアルゴリズムを使用して API リクエスト間のスリープ間隔を計算することをお勧めします。エクスポネンシャルバックオフの背後にある考え方は、連続したエラー応答の再試行間の待機時間を徐々に長く使用することです。最大遅延間隔と最大再試行回数を実装する必要があります。ジッター (ランダム化された遅延) を使用して、連続する衝突を防ぐこともできます。詳細については、「[タイムアウト、リトライ、ジッターによるバックオフ](#)」を参照してください。

各 AWS SDK は自動再試行ロジックを実装します。詳細については、「AWS SDK とツールのリファレンスガイド」の「[再試行動作](#)」を参照してください。

制限の引き上げのリクエスト

AWS アカウントの API スロットリング上限の引き上げをリクエストできます。

レコメンデーション

- 1 つのリクエストで既存の上限の最大 3 倍をリクエストします。
- バケットの最大容量を増やす前に、バケットリフィルレートの増加を優先します。
- リクエストされたバケットリフィルレートがバケットの最大容量を超える場合は、バケットの最大容量を同時に増やします。
- 増加が必要なすべての API アクションを指定します。上限は、API アクションのカテゴリではなく、個々の API アクションに適用されます。
- RunInstances、StartInstances、StopInstances、TerminateInstances の API アクションにはリクエストレートとリソースレートの両方の上限があります。引き上げる上限を必ず指定してください。

この機能へのアクセスをリクエストするには

1. [AWS サポート センター](#)を開きます。
2. [ケースを作成] を選択します。
3. [Account and billing] (アカウントおよび請求) を選択します。
4. [サービス] で [一般情報と開始方法] を選択します。
5. カテゴリ で、「Using AWS & Services」を選択します。
6. [次のステップ: 追加情報] を選択します。

7. [Subject (件名)] に **Request an increase in my Amazon EC2 API throttling limits** と入力します。
8. [説明] には、次のテンプレートをコピーし、必要な情報を入力します。

```
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. [次のステップ: 今すぐ解決するか、お問い合わせください] を選択します。
10. [お問い合わせ] タブをクリックし、連絡の際に希望する言語と連絡方法を選択します。
11. [Submit] を選択します。

Amazon EC2 API でのページ分割

DescribeInstances など、多数の結果を返す可能性のある記述アクションを呼び出す場合は、ページ分割を使用することをお勧めします。ページ分割を使用すると、記述呼び出しによって返される項目の数と、呼び出しが戻るのにかかる時間が制限されます。多数のリソースがある場合、ページ分割されていない呼び出しはスロットリングされ、タイムアウトする可能性があります。したがって、ページ分割された呼び出しは一貫して成功するため、ページ分割されていない呼び出しよりもページ分割された呼び出しの方が全体的なレイテンシーが優れています。

詳細については、「Amazon EC2 API リファレンス」の「[Pagination](#)」(ページネーション)を参照してください。

ベストプラクティス

可能な場合は、記述呼び出しでリソース ID のリストを指定します。これは、多数のリソースを記述する最も速い方法です。1 回の呼び出しで 1,000 を超える ID を指定しないでください。以下に例を示します。

```
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();
}
```

記述呼び出しでリソース ID を指定できない場合は、ページ分割を使用することを強くお勧めします。以下に例を示します。

```
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;
}
```

ページ分割された呼び出しを再試行する必要がある場合は、[ジッターでエクスポネンシャルバックオフ](#)を使用します。

一般的な問題

以下は、誤ってページ分割されていない呼び出しを行うコードの例です。

Example問題の例: リソース ID の空のリストを渡す

次のコードは ID のリストを使用します。ただし、リストが空の場合、結果はページ分割されていない呼び出しになります。

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

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

この問題を修正するには、記述呼び出しを行う前にリストが空でないことを確認してください。

```
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問題の例: MaxResults を設定しない

次のコードは nextToken をチェックして使用しますが、MaxResults を設定しません。

```
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;
}
```

この問題を修正するには、次のように withMaxResults を追加します。

```
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;
}
```

AWS CLI を使用して Amazon EC2 リソースを作成する

Amazon EC2 リソースを作成および管理するには、コマンドラインシェルの AWS Command Line Interface (AWS CLI) を使用します。AWS CLI は、Amazon EC2 などの AWS のサービスの API に直接アクセスできます。

Amazon EC2 のコマンドの構文と例については、「AWS CLI コマンドリファレンス」の「[ec2](#)」を参照してください。これらの例は、GitHub の [aws-cli/awscli/examples/ec2](https://awscli.amazonaws.com/awscli-examples/ec2) でも確認できます。

AWS CLI の詳細

AWS CLI の詳細については、以下のリソースを参照してください。

- [AWS Command Line Interface](#)
- [AWS Command Line Interface バージョン 2 用ユーザーガイド](#)
- [AWS Command Line Interface バージョン 1 用ユーザーガイド](#)

AWS CloudFormation を使用して Amazon EC2 リソースを作成する

Amazon EC2 は AWS CloudFormation と統合されています。これは、リソースとインフラストラクチャの作成と管理の所要時間を短縮できるように AWS リソースをモデル化して設定するためのサービスです。必要な AWS リソース (インスタンスやサブネットなど) を説明するテンプレートを作成すると、CloudFormation がユーザーに代わってこれらのリソースのセットアップや構成を行います。

CloudFormation を使用すると、テンプレートを再利用して Amazon EC2 リソースをいつでも繰り返しセットアップできます。リソースを一度記述するだけで、同じリソースを複数の AWS アカウントとリージョンで何度でもプロビジョニングできます。

Amazon EC2 と CloudFormation テンプレート

Amazon EC2 および関連サービスのリソースをプロビジョニングして設定するには、[CloudFormation テンプレート](#)を理解しておく必要があります。テンプレートは、JSON や YAML でフォーマットされたテキストファイルです。これらのテンプレートには、CloudFormation スタックにプロビジョニングするリソースを記述します。JSON や YAML に不慣れな方は、CloudFormation デザイナー を使えば、CloudFormation テンプレートを使いこなすことができます。詳細については、「AWS CloudFormation ユーザーガイド」の「[CloudFormation Designer とは](#)」を参照してください。

Amazon EC2 のリソース

コンピューティングリソース

- [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](#)

ネットワーキングリソース

- [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](#)

セキュリティリソース

- [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](#)

ストレージリソース

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

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

CloudFormation の詳細はこちら

CloudFormation の詳細については、以下のリソースを参照してください。

- [AWS CloudFormation](#)
- [AWS CloudFormation ユーザーガイド](#)

AWS SDK を使用して Amazon EC2 リソースを作成する

AWS は、多くの一般的なプログラミング言語で利用できる Software Development Kit (SDK) を提供します。SDK は、以下を提供することで開発をより効率的にします。

- アプリケーションに組み込むことができる構築済みのコンポーネントとライブラリ
- コンパイラやデバッガーなどの言語固有のツール
- サービスリクエストの暗号化署名
- リクエストの再試行
- エラーレスポンスの処理

Amazon EC2 API のコードサンプル

AWS が提供するコードサンプルは、API を使用して特定のタスクを実行する方法を示しています。Amazon EC2 API の例については、[Amazon EC2 のコードサンプル](#)」を参照してください。その他の例については、「[AWS SDK のコードサンプルを検索する](#)」または GitHub の「[aws-doc-sdk-examples](#)」を参照してください。

AWS SDK の詳細

AWS SDK の詳細については、以下のリソースを参照してください。

- [AWS SDK とツールのリファレンスガイド](#)
- [AWS に構築するツール](#)
- [SDK とは](#)

Amazon EC2 の低レベル API

Amazon EC2 の低レベル API は、Amazon EC2 用のプロトコルレベルのインターフェイスです。低レベル API を使用する場合は、すべての HTTPS リクエストを正しくフォーマットし、すべてのリクエストに有効なデジタル署名を追加する必要があります。詳細については、「Amazon EC2 API リファレンス」の「[Amazon EC2 API へのリクエストを作成する](#)」を参照してください。または、ユーザーに代わってリクエストを作成して署名する AWS SDK を使用することもできます。詳細については、「[AWS SDK を使用する](#)」を参照してください。

Amazon EC2 API は、複数のサービスのアクションとデータ型で構成されます。各サービスのアクションを確認するには、「Amazon EC2 API リファレンス」の以下のページを参照してください。

- [AWS Client VPN アクション](#)
- [Amazon EBS アクション](#)
- [Amazon EC2 アクション](#)
- [AWS Network Manager アクション](#)
- [AWS Nitro Enclaves アクション](#)
- [AWS Outposts アクション](#)
- [AWS PrivateLink アクション](#)
- [ごみ箱アクション](#)
- [AWS Site-to-Site VPNActions](#)
- [AWS Transit Gateway アクション](#)
- [AWS Verified Access アクション](#)
- [VM Import/Export アクション](#)
- [Amazon VPC アクション](#)
- [Amazon VPC IPAM アクション](#)
- [AWS Wavelength アクション](#)

Console-to-Code を使用して EC2 コンソールアクションのコードを生成する

コンソールには、リソースの作成とプロトタイプの実行を行うためのガイド付きのパスが用意されています。同じリソースを大規模に作成したい場合は、自動化コードが必要です。Console-to-Code は Amazon Q Developer の機能であり、オートメーションコードの使用を開始するために役立ちます。Console-to-Code は、デフォルト設定や互換性のあるパラメータを含むコンソールのアクションを記録します。次に、生成 AI を使用して、必要なアクションに応じて、推奨される Infrastructure as Code (IaC) 形式のコードを提案します。コンソールワークフローは指定されたパラメータ値の併用が有効であることを確実にするため、Console-to-Code を使用して生成されたコードには、互換性のあるパラメータ値があります。このコードを出発点として使用し、特定のユースケースの本番環境に対応するようにカスタマイズできます。

例えば、Console-to-Code では、Amazon EC2 インスタンスの起動を記録し、CloudFormation JSON 形式でコードを生成する選択ができます。その後、そのコードをコピーして、CloudFormation テンプレートでの使用向けにカスタマイズできます。

現在、Console-to-Code は次の言語と形式で Infrastructure as Code (IaC) を生成できます。

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

Console-to-Code の使用方法に関する詳細と手順については、[「Amazon Q Developer User Guide」の「Automating AWS services with Amazon Q Developer Console-to-Code」](#)を参照してください。

SDK を使用した Amazon EC2 のコード例 AWS SDKs

次のコード例は、AWS Software Development Kit (SDK) で Amazon EC2 を使用方法を示しています。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

シナリオは、1つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

コードの例

- [SDK を使用した Amazon EC2 の基本的な例 AWS SDKs](#)
 - [Hello Amazon EC2](#)
 - [AWS SDK を使用した Amazon EC2 の基本について説明します。](#)
 - [SDK を使用した Amazon EC2 のアクション AWS SDKs](#)
 - [CLI で AcceptVpcPeeringConnection を使用する](#)
 - [AWS SDK または CLI AllocateAddress で使用する](#)
 - [CLI で AllocateHosts を使用する](#)
 - [CLI で AssignPrivateIpAddresses を使用する](#)
 - [AWS SDK または CLI AssociateAddress で使用する](#)
 - [CLI で AssociateDhcpOptions を使用する](#)
 - [CLI で AssociateRouteTable を使用する](#)
 - [CLI で AttachInternetGateway を使用する](#)
 - [CLI で AttachNetworkInterface を使用する](#)
 - [CLI で AttachVolume を使用する](#)
 - [CLI で AttachVpnGateway を使用する](#)
 - [CLI で AuthorizeSecurityGroupEgress を使用する](#)

- [AWS SDK または CLI AuthorizeSecurityGroupIngressで を使用する](#)
- [CLI で CancelCapacityReservation を使用する](#)
- [CLI で CancellImportTask を使用する](#)
- [CLI で CancelSpotFleetRequests を使用する](#)
- [CLI で CancelSpotInstanceRequests を使用する](#)
- [CLI で ConfirmProductInstance を使用する](#)
- [CLI で CopyImage を使用する](#)
- [CLI で CopySnapshot を使用する](#)
- [CLI で CreateCapacityReservation を使用する](#)
- [CLI で CreateCustomerGateway を使用する](#)
- [CLI で CreateDhcpOptions を使用する](#)
- [CLI で CreateFlowLogs を使用する](#)
- [CLI で CreateImage を使用する](#)
- [CLI で CreateInstanceExportTask を使用する](#)
- [CLI で CreateInternetGateway を使用する](#)
- [AWS SDK または CLI CreateKeyPairで を使用する](#)
- [AWS SDK または CLI CreateLaunchTemplateで を使用する](#)
- [CLI で CreateNetworkAcl を使用する](#)
- [CLI で CreateNetworkAclEntry を使用する](#)
- [CLI で CreateNetworkInterface を使用する](#)
- [CLI で CreatePlacementGroup を使用する](#)
- [CLI で CreateRoute を使用する](#)
- [AWS SDK または CLI CreateRouteTableで を使用する](#)
- [AWS SDK または CLI CreateSecurityGroupで を使用する](#)
- [CLI で CreateSnapshot を使用する](#)
- [CLI で CreateSpotDatafeedSubscription を使用する](#)
- [AWS SDK または CLI CreateSubnetで を使用する](#)
- [AWS SDK または CLI CreateTagsで を使用する](#)
- [CLI で CreateVolume を使用する](#)
- [AWS SDK または CLI CreateVpcで を使用する](#)

- [AWS SDK または CLI CreateVpcEndpointで を使用する](#)
- [CLI で CreateVpnConnection を使用する](#)
- [CLI で CreateVpnConnectionRoute を使用する](#)
- [CLI で CreateVpnGateway を使用する](#)
- [CLI で DeleteCustomerGateway を使用する](#)
- [CLI で DeleteDhcpOptions を使用する](#)
- [CLI で DeleteFlowLogs を使用する](#)
- [CLI で DeleteInternetGateway を使用する](#)
- [AWS SDK または CLI DeleteKeyPairで を使用する](#)
- [AWS SDK または CLI DeleteLaunchTemplateで を使用する](#)
- [CLI で DeleteNetworkAcl を使用する](#)
- [CLI で DeleteNetworkAclEntry を使用する](#)
- [CLI で DeleteNetworkInterface を使用する](#)
- [CLI で DeletePlacementGroup を使用する](#)
- [CLI で DeleteRoute を使用する](#)
- [CLI で DeleteRouteTable を使用する](#)
- [AWS SDK または CLI DeleteSecurityGroupで を使用する](#)
- [AWS SDK または CLI DeleteSnapshotで を使用する](#)
- [CLI で DeleteSpotDatafeedSubscription を使用する](#)
- [CLI で DeleteSubnet を使用する](#)
- [CLI で DeleteTags を使用する](#)
- [CLI で DeleteVolume を使用する](#)
- [AWS SDK または CLI DeleteVpcで を使用する](#)
- [AWS SDK または CLI DeleteVpcEndpointsで を使用する](#)
- [CLI で DeleteVpnConnection を使用する](#)
- [CLI で DeleteVpnConnectionRoute を使用する](#)
- [CLI で DeleteVpnGateway を使用する](#)
- [CLI で DeregisterImage を使用する](#)
- [CLI で DescribeAccountAttributes を使用する](#)
- [AWS SDK または CLI DescribeAddressesで を使用する](#)

- [AWS SDK または CLI DescribeAvailabilityZonesで を使用する](#)
- [CLI で DescribeBundleTasks を使用する](#)
- [CLI で DescribeCapacityReservations を使用する](#)
- [CLI で DescribeCustomerGateways を使用する](#)
- [CLI で DescribeDhcpOptions を使用する](#)
- [CLI で DescribeFlowLogs を使用する](#)
- [CLI で DescribeHostReservationOfferings を使用する](#)
- [CLI で DescribeHosts を使用する](#)
- [AWS SDK または CLI DescribelamInstanceProfileAssociationsで を使用する](#)
- [CLI で DescribelFormat を使用する](#)
- [CLI で DescribelIdentityIdFormat を使用する](#)
- [CLI で DescribelImageAttribute を使用する](#)
- [AWS SDK または CLI DescribelImagesで を使用する](#)
- [CLI で DescribelImportImageTasks を使用する](#)
- [CLI で DescribelImportSnapshotTasks を使用する](#)
- [CLI で DescribelInstanceAttribute を使用する](#)
- [AWS SDK または CLI DescribelInstanceStatusで を使用する](#)
- [AWS SDK または CLI DescribelInstanceTypesで を使用する](#)
- [AWS SDK または CLI DescribelInstancesで を使用する](#)
- [CLI で DescribelInternetGateways を使用する](#)
- [AWS SDK または CLI DescribeKeyPairsで を使用する](#)
- [CLI で DescribeNetworkAcls を使用する](#)
- [CLI で DescribeNetworkInterfaceAttribute を使用する](#)
- [CLI で DescribeNetworkInterfaces を使用する](#)
- [CLI で DescribePlacementGroups を使用する](#)
- [CLI で DescribePrefixLists を使用する](#)
- [AWS SDK または CLI DescribeRegionsで を使用する](#)
- [AWS SDK または CLI DescribeRouteTablesで を使用する](#)
- [CLI で DescribeScheduledInstanceAvailability を使用する](#)
- [CLI で DescribeScheduledInstances を使用する](#)

- [AWS SDK または CLI DescribeSecurityGroups で 使用する](#)
- [CLI で DescribeSnapshotAttribute を使用する](#)
- [AWS SDK または CLI DescribeSnapshots で 使用する](#)
- [CLI で DescribeSpotDatafeedSubscription を使用する](#)
- [CLI で DescribeSpotFleetInstances を使用する](#)
- [CLI で DescribeSpotFleetRequestHistory を使用する](#)
- [CLI で DescribeSpotFleetRequests を使用する](#)
- [CLI で DescribeSpotInstanceRequests を使用する](#)
- [CLI で DescribeSpotPriceHistory を使用する](#)
- [AWS SDK または CLI DescribeSubnets で 使用する](#)
- [CLI で DescribeTags を使用する](#)
- [CLI で DescribeVolumeAttribute を使用する](#)
- [CLI で DescribeVolumeStatus を使用する](#)
- [CLI で DescribeVolumes を使用する](#)
- [CLI で DescribeVpcAttribute を使用する](#)
- [CLI で DescribeVpcClassicLink を使用する](#)
- [CLI で DescribeVpcClassicLinkDnsSupport を使用する](#)
- [CLI で DescribeVpcEndpointServices を使用する](#)
- [CLI で DescribeVpcEndpoints を使用する](#)
- [AWS SDK または CLI DescribeVpcs で 使用する](#)
- [CLI で DescribeVpnConnections を使用する](#)
- [CLI で DescribeVpnGateways を使用する](#)
- [CLI で DetachInternetGateway を使用する](#)
- [CLI で DetachNetworkInterface を使用する](#)
- [CLI で DetachVolume を使用する](#)
- [CLI で DetachVpnGateway を使用する](#)
- [CLI で DisableVgwRoutePropagation を使用する](#)
- [CLI で DisableVpcClassicLink を使用する](#)
- [CLI で DisableVpcClassicLinkDnsSupport を使用する](#)
- [AWS SDK または CLI DisassociateAddress で 使用する](#)

- [CLI で DisassociateRouteTable を使用する](#)
- [CLI で EnableVgwRoutePropagation を使用する](#)
- [CLI で EnableVolumelo を使用する](#)
- [CLI で EnableVpcClassicLink を使用する](#)
- [CLI で EnableVpcClassicLinkDnsSupport を使用する](#)
- [CLI で GetConsoleOutput を使用する](#)
- [CLI で GetHostReservationPurchasePreview を使用する](#)
- [AWS SDK または CLI GetPasswordDataで を使用する](#)
- [CLI で ImportImage を使用する](#)
- [CLI で ImportKeyPair を使用する](#)
- [CLI で ImportSnapshot を使用する](#)
- [CLI で ModifyCapacityReservation を使用する](#)
- [CLI で ModifyHosts を使用する](#)
- [CLI で ModifyIdFormat を使用する](#)
- [CLI で ModifyImageAttribute を使用する](#)
- [CLI で ModifyInstanceAttribute を使用する](#)
- [CLI で ModifyInstanceCreditSpecification を使用する](#)
- [CLI で ModifyNetworkInterfaceAttribute を使用する](#)
- [CLI で ModifyReservedInstances を使用する](#)
- [CLI で ModifySnapshotAttribute を使用する](#)
- [CLI で ModifySpotFleetRequest を使用する](#)
- [CLI で ModifySubnetAttribute を使用する](#)
- [CLI で ModifyVolumeAttribute を使用する](#)
- [CLI で ModifyVpcAttribute を使用する](#)
- [AWS SDK または CLI MonitorInstancesで を使用する](#)
- [CLI で MoveAddressToVpc を使用する](#)
- [CLI で PurchaseHostReservation を使用する](#)
- [CLI で PurchaseScheduledInstances を使用する](#)
- [AWS SDK または CLI RebootInstancesで を使用する](#)
- [CLI で RegisterImage を使用する](#)

- [CLI で RejectVpcPeeringConnection を使用する](#)
- [AWS SDK または CLI ReleaseAddress で使用する](#)
- [CLI で ReleaseHosts を使用する](#)
- [AWS SDK または CLI ReplaceIamInstanceProfileAssociation で使用する](#)
- [CLI で ReplaceNetworkAclAssociation を使用する](#)
- [CLI で ReplaceNetworkAclEntry を使用する](#)
- [CLI で ReplaceRoute を使用する](#)
- [CLI で ReplaceRouteTableAssociation を使用する](#)
- [CLI で ReportInstanceState を使用する](#)
- [CLI で RequestSpotFleet を使用する](#)
- [CLI で RequestSpotInstances を使用する](#)
- [CLI で ResetImageAttribute を使用する](#)
- [CLI で ResetInstanceAttribute を使用する](#)
- [CLI で ResetNetworkInterfaceAttribute を使用する](#)
- [CLI で ResetSnapshotAttribute を使用する](#)
- [CLI で RevokeSecurityGroupEgress を使用する](#)
- [CLI で RevokeSecurityGroupIngress を使用する](#)
- [AWS SDK または CLI RunInstances で使用する](#)
- [CLI で RunScheduledInstances を使用する](#)
- [AWS SDK または CLI StartInstances で使用する](#)
- [AWS SDK または CLI StopInstances で使用する](#)
- [AWS SDK または CLI TerminateInstances で使用する](#)
- [CLI で UnassignPrivateIpAddresses を使用する](#)
- [AWS SDK または CLI UnmonitorInstances で使用する](#)
- [CLI で UpdateSecurityGroupRuleDescriptionsIngress を使用する](#)
- [SDK を使用した Amazon EC2 のシナリオ AWS SDKs](#)
 - [AWS SDK を使用して回復力のあるサービスを構築および管理します。](#)
 - [CLI を使用してプライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成します。](#)
- [CLI を使用した Amazon VPC の開始方法](#)
- [CLI を使用した Transit Gateway の開始方法](#)

- [CLI を使用した Amazon VPC IPAM の開始方法](#)

SDK を使用した Amazon EC2 の基本的な例 AWS SDKs

次のコード例では、AWS SDK で Amazon Elastic Compute Cloud を使用する基本的な方法を説明しています。

例

- [Hello Amazon EC2](#)
- [AWS SDK を使用した Amazon EC2 の基本について説明します。](#)
- [SDK を使用した Amazon EC2 のアクション AWS SDKs](#)
 - [CLI で AcceptVpcPeeringConnection を使用する](#)
 - [AWS SDK または CLI AllocateAddress で 使用する](#)
 - [CLI で AllocateHosts を使用する](#)
 - [CLI で AssignPrivateIpAddresses を使用する](#)
 - [AWS SDK または CLI AssociateAddress で 使用する](#)
 - [CLI で AssociateDhcpOptions を使用する](#)
 - [CLI で AssociateRouteTable を使用する](#)
 - [CLI で AttachInternetGateway を使用する](#)
 - [CLI で AttachNetworkInterface を使用する](#)
 - [CLI で AttachVolume を使用する](#)
 - [CLI で AttachVpnGateway を使用する](#)
 - [CLI で AuthorizeSecurityGroupEgress を使用する](#)
 - [AWS SDK または CLI AuthorizeSecurityGroupIngress で 使用する](#)
 - [CLI で CancelCapacityReservation を使用する](#)
 - [CLI で CancellImportTask を使用する](#)
 - [CLI で CancelSpotFleetRequests を使用する](#)
 - [CLI で CancelSpotInstanceRequests を使用する](#)
 - [CLI で ConfirmProductInstance を使用する](#)
 - [CLI で CopyImage を使用する](#)
 - 基本 [CLI で CopySnapshot を使用する](#)

- [CLI で CreateCapacityReservation を使用する](#)
- [CLI で CreateCustomerGateway を使用する](#)
- [CLI で CreateDhcpOptions を使用する](#)
- [CLI で CreateFlowLogs を使用する](#)
- [CLI で CreateImage を使用する](#)
- [CLI で CreateInstanceExportTask を使用する](#)
- [CLI で CreateInternetGateway を使用する](#)
- [AWS SDK または CLI CreateKeyPairで を使用する](#)
- [AWS SDK または CLI CreateLaunchTemplateで を使用する](#)
- [CLI で CreateNetworkAcl を使用する](#)
- [CLI で CreateNetworkAclEntry を使用する](#)
- [CLI で CreateNetworkInterface を使用する](#)
- [CLI で CreatePlacementGroup を使用する](#)
- [CLI で CreateRoute を使用する](#)
- [AWS SDK または CLI CreateRouteTableで を使用する](#)
- [AWS SDK または CLI CreateSecurityGroupで を使用する](#)
- [CLI で CreateSnapshot を使用する](#)
- [CLI で CreateSpotDatafeedSubscription を使用する](#)
- [AWS SDK または CLI CreateSubnetで を使用する](#)
- [AWS SDK または CLI CreateTagsで を使用する](#)
- [CLI で CreateVolume を使用する](#)
- [AWS SDK または CLI CreateVpcで を使用する](#)
- [AWS SDK または CLI CreateVpcEndpointで を使用する](#)
- [CLI で CreateVpnConnection を使用する](#)
- [CLI で CreateVpnConnectionRoute を使用する](#)
- [CLI で CreateVpnGateway を使用する](#)
- [CLI で DeleteCustomerGateway を使用する](#)
- [CLI で DeleteDhcpOptions を使用する](#)
- [CLI で DeleteFlowLogs を使用する](#)
- [CLI で DeleteInternetGateway を使用する](#)

- [AWS SDK または CLI DeleteKeyPairで を使用する](#)
- [AWS SDK または CLI DeleteLaunchTemplateで を使用する](#)
- [CLI で DeleteNetworkAcl を使用する](#)
- [CLI で DeleteNetworkAclEntry を使用する](#)
- [CLI で DeleteNetworkInterface を使用する](#)
- [CLI で DeletePlacementGroup を使用する](#)
- [CLI で DeleteRoute を使用する](#)
- [CLI で DeleteRouteTable を使用する](#)
- [AWS SDK または CLI DeleteSecurityGroupで を使用する](#)
- [AWS SDK または CLI DeleteSnapshotで を使用する](#)
- [CLI で DeleteSpotDatafeedSubscription を使用する](#)
- [CLI で DeleteSubnet を使用する](#)
- [CLI で DeleteTags を使用する](#)
- [CLI で DeleteVolume を使用する](#)
- [AWS SDK または CLI DeleteVpcで を使用する](#)
- [AWS SDK または CLI DeleteVpcEndpointsで を使用する](#)
- [CLI で DeleteVpnConnection を使用する](#)
- [CLI で DeleteVpnConnectionRoute を使用する](#)
- [CLI で DeleteVpnGateway を使用する](#)
- [CLI で DeregisterImage を使用する](#)
- [CLI で DescribeAccountAttributes を使用する](#)
- [AWS SDK または CLI DescribeAddressesで を使用する](#)
- [AWS SDK または CLI DescribeAvailabilityZonesで を使用する](#)
- [CLI で DescribeBundleTasks を使用する](#)
- [CLI で DescribeCapacityReservations を使用する](#)
- [CLI で DescribeCustomerGateways を使用する](#)
- [CLI で DescribeDhcpOptions を使用する](#)
- [CLI で DescribeFlowLogs を使用する](#)
- [CLI で DescribeHostReservationOfferings を使用する](#)
- [CLI で DescribeHosts を使用する](#)

- [AWS SDK または CLI DescribeInstanceProfileAssociations で使用する](#)
- [CLI で DescribeIdFormat を使用する](#)
- [CLI で DescribeIdentityIdFormat を使用する](#)
- [CLI で DescribeImageAttribute を使用する](#)
- [AWS SDK または CLI DescribeImages で使用する](#)
- [CLI で DescribeImportImageTasks を使用する](#)
- [CLI で DescribeImportSnapshotTasks を使用する](#)
- [CLI で DescribeInstanceAttribute を使用する](#)
- [AWS SDK または CLI DescribeInstanceStatus で使用する](#)
- [AWS SDK または CLI DescribeInstanceTypes で使用する](#)
- [AWS SDK または CLI DescribeInstances で使用する](#)
- [CLI で DescribeInternetGateways を使用する](#)
- [AWS SDK または CLI DescribeKeyPairs で使用する](#)
- [CLI で DescribeNetworkAcls を使用する](#)
- [CLI で DescribeNetworkInterfaceAttribute を使用する](#)
- [CLI で DescribeNetworkInterfaces を使用する](#)
- [CLI で DescribePlacementGroups を使用する](#)
- [CLI で DescribePrefixLists を使用する](#)
- [AWS SDK または CLI DescribeRegions で使用する](#)
- [AWS SDK または CLI DescribeRouteTables で使用する](#)
- [CLI で DescribeScheduledInstanceAvailability を使用する](#)
- [CLI で DescribeScheduledInstances を使用する](#)
- [AWS SDK または CLI DescribeSecurityGroups で使用する](#)
- [CLI で DescribeSnapshotAttribute を使用する](#)
- [AWS SDK または CLI DescribeSnapshots で使用する](#)
- [CLI で DescribeSpotDatafeedSubscription を使用する](#)
- [CLI で DescribeSpotFleetInstances を使用する](#)
- [CLI で DescribeSpotFleetRequestHistory を使用する](#)
- [CLI で DescribeSpotFleetRequests を使用する](#)
- [CLI で DescribeSpotInstanceRequests を使用する](#)

- [CLI で DescribeSpotPriceHistory を使用する](#)
- [AWS SDK または CLI DescribeSubnets で使用する](#)
- [CLI で DescribeTags を使用する](#)
- [CLI で DescribeVolumeAttribute を使用する](#)
- [CLI で DescribeVolumeStatus を使用する](#)
- [CLI で DescribeVolumes を使用する](#)
- [CLI で DescribeVpcAttribute を使用する](#)
- [CLI で DescribeVpcClassicLink を使用する](#)
- [CLI で DescribeVpcClassicLinkDnsSupport を使用する](#)
- [CLI で DescribeVpcEndpointServices を使用する](#)
- [CLI で DescribeVpcEndpoints を使用する](#)
- [AWS SDK または CLI DescribeVpcs で使用する](#)
- [CLI で DescribeVpnConnections を使用する](#)
- [CLI で DescribeVpnGateways を使用する](#)
- [CLI で DetachInternetGateway を使用する](#)
- [CLI で DetachNetworkInterface を使用する](#)
- [CLI で DetachVolume を使用する](#)
- [CLI で DetachVpnGateway を使用する](#)
- [CLI で DisableVgwRoutePropagation を使用する](#)
- [CLI で DisableVpcClassicLink を使用する](#)
- [CLI で DisableVpcClassicLinkDnsSupport を使用する](#)
- [AWS SDK または CLI DisassociateAddress で使用する](#)
- [CLI で DisassociateRouteTable を使用する](#)
- [CLI で EnableVgwRoutePropagation を使用する](#)
- [CLI で EnableVolumelo を使用する](#)
- [CLI で EnableVpcClassicLink を使用する](#)
- [CLI で EnableVpcClassicLinkDnsSupport を使用する](#)
- [CLI で GetConsoleOutput を使用する](#)
- [CLI で GetHostReservationPurchasePreview を使用する](#)
- [AWS SDK または CLI GetPasswordData で使用する](#)

- [CLI で ImportImage を使用する](#)
- [CLI で ImportKeyPair を使用する](#)
- [CLI で ImportSnapshot を使用する](#)
- [CLI で ModifyCapacityReservation を使用する](#)
- [CLI で ModifyHosts を使用する](#)
- [CLI で ModifyIdFormat を使用する](#)
- [CLI で ModifyImageAttribute を使用する](#)
- [CLI で ModifyInstanceAttribute を使用する](#)
- [CLI で ModifyInstanceCreditSpecification を使用する](#)
- [CLI で ModifyNetworkInterfaceAttribute を使用する](#)
- [CLI で ModifyReservedInstances を使用する](#)
- [CLI で ModifySnapshotAttribute を使用する](#)
- [CLI で ModifySpotFleetRequest を使用する](#)
- [CLI で ModifySubnetAttribute を使用する](#)
- [CLI で ModifyVolumeAttribute を使用する](#)
- [CLI で ModifyVpcAttribute を使用する](#)
- [AWS SDK または CLI MonitorInstancesで を使用する](#)
- [CLI で MoveAddressToVpc を使用する](#)
- [CLI で PurchaseHostReservation を使用する](#)
- [CLI で PurchaseScheduledInstances を使用する](#)
- [AWS SDK または CLI RebootInstancesで を使用する](#)
- [CLI で RegisterImage を使用する](#)
- [CLI で RejectVpcPeeringConnection を使用する](#)
- [AWS SDK または CLI ReleaseAddressで を使用する](#)
- [CLI で ReleaseHosts を使用する](#)
- [AWS SDK または CLI ReplaceIamInstanceProfileAssociationで を使用する](#)
- [CLI で ReplaceNetworkAclAssociation を使用する](#)
- [CLI で ReplaceNetworkAclEntry を使用する](#)
- [CLI で ReplaceRoute を使用する](#)
- [CLI で ReplaceRouteTableAssociation を使用する](#)

- [CLI で ReportInstanceStatus を使用する](#)
- [CLI で RequestSpotFleet を使用する](#)
- [CLI で RequestSpotInstances を使用する](#)
- [CLI で ResetImageAttribute を使用する](#)
- [CLI で ResetInstanceAttribute を使用する](#)
- [CLI で ResetNetworkInterfaceAttribute を使用する](#)
- [CLI で ResetSnapshotAttribute を使用する](#)
- [CLI で RevokeSecurityGroupEgress を使用する](#)
- [CLI で RevokeSecurityGroupIngress を使用する](#)
- [AWS SDK または CLI RunInstancesで を使用する](#)
- [CLI で RunScheduledInstances を使用する](#)
- [AWS SDK または CLI StartInstancesで を使用する](#)
- [AWS SDK または CLI StopInstancesで を使用する](#)
- [AWS SDK または CLI TerminateInstancesで を使用する](#)
- [CLI で UnassignPrivateIpAddresses を使用する](#)
- [AWS SDK または CLI UnmonitorInstancesで を使用する](#)
- [CLI で UpdateSecurityGroupRuleDescriptionsIngress を使用する](#)

Hello Amazon EC2

次のコード例は、Amazon EC2 の使用を開始する方法を示しています。

.NET

SDK for .NET (v4)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
namespace EC2Actions;
```

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

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

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

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

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

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

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

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[DescribeSecurityGroups](#)」を参照してください。

C++

SDK for C++

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

CMakeLists.txt CMake ファイルのコード。

```
# 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 ソースファイルのコード。

```

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

```

```
#include <iostream>

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

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

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

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

```

    }

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

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

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

            Aws::String typeString =

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

            Aws::String monitorString =

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

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

```

```
        }

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

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

- APIの詳細については、「AWS SDK for C++ API リファレンス」の「[DescribeSecurityGroups](#)」を参照してください。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/**
 * Asynchronously describes the security groups for the specified group ID.
 *
 * @param groupName the name of the security group to describe
```

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

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

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[DescribeSecurityGroups](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

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

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

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

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

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[DescribeSecurityGroups](#)」を参照してください。

Kotlin

SDK for Kotlin

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[DescribeSecurityGroups](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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"))
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[DescribeSecurityGroups](#)」を参照してください。

Ruby

SDK for Ruby

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

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

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

    instances = fetch_instances

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

  private

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

```
instances = []

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

instances

end

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

end

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

- APIの詳細については、AWS SDK for Ruby API リファレンスの「[DescribeSecurityGroups](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

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

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[DescribeSecurityGroups](#)」を参照してください。

Swift

SDK for Swift

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

Package.swift ファイル。

```
// 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 ファイル。

```

// 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)
        }
    }
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[DescribeSecurityGroups](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用した Amazon EC2 の基本について説明します。

次のコード例は、以下を実行する方法を示しています。

- キーペアとセキュリティグループを作成します。
- Amazon マシンイメージ (AMI) と互換性のあるインスタンスタイプを選択し、インスタンスを作成します。
- インスタンスを停止し、再起動します。
- Elastic IP アドレスをインスタンスに関連付けます。
- SSH を使用してインスタンスに接続し、リソースをクリーンアップします。

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

コマンドプロンプトでシナリオを実行します。

```
/// <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 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.Write("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($"\\t{i++}\\t{image.Description}");
        });

        int choice = 1;
        bool validNumber = false;
        if (isInteractive)
```

```
{
    do
    {
        Console.WriteLine("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);
    }
}

```

EC2 アクションをラップするクラスを定義します。

```

/// <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 {groupName} does not exist and
cannot be deleted. Please verify the group name and try again.");
        }
    }
}
```

```
        {
            _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($"\\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.Write(". ");
    } while (!hasState);

    return hasState;
}
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の以下のトピックを参照してください。
 - [AllocateAddress](#)
 - [AssociateAddress](#)
 - [AuthorizeSecurityGroupIngress](#)
 - [CreateKeyPair](#)
 - [CreateSecurityGroup](#)
 - [DeleteKeyPair](#)
 - [DeleteSecurityGroup](#)
 - [DescribeImages](#)
 - [DescribeInstanceTypes](#)
 - [DescribeInstances](#)
 - [DescribeKeyPairs](#)
 - [DescribeSecurityGroups](#)
 - [DisassociateAddress](#)
 - [ReleaseAddress](#)
 - [RunInstances](#)
 - [StartInstances](#)
 - [StopInstances](#)
 - [TerminateInstances](#)
 - [UnmonitorInstances](#)

Bash

AWS CLI Bash スクリプトを使用する

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブなシナリオを実行します。

```
#####
# 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

```

```
    }
fi

echo_repeat "*" 88
echo_repeat "*" 88

echo "Let's create a security group to manage access to your instance."
echo -n "Enter a unique name for your security group: "
get_input
local security_group_name
security_group_name=$get_input_result
local security_group_id
security_group_id=$(ec2_create_security_group -n "$security_group_name" \
    -d "Security group for EC2 instance") || {
    errecho "The security failed to create. This demo will exit."
    clean_up "$key_name" "$key_file_name"
    return 1
}

echo "Security group created with ID $security_group_id"
echo

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

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

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

echo "Security group rules updated"

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

```

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

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

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

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

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

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

```

```

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

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

echo_repeat "*" 88
echo_repeat "*" 88

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

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

echo "Great choice."
echo

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

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

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

```



```

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

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

echo "Creating your instance and waiting for it to start..."
local instance_id
instance_id=$(ec2_run_instances -i "$image_id" -t "$instance_type" -k
"$key_name" -s "$security_group_id") || {
    errecho "Failed to run instance. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1
}

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

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

echo
print_instance_details "${instance_details}"

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

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

echo_repeat "*" 88

```

```
echo_repeat "*" 88

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

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

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

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

print_instance_details "${instance_details}"

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

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

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

echo_repeat "*" 88
echo_repeat "*" 88

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

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

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

```
echo "Allocated static Elastic IP address: $elastic_ip"

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

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

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

echo_repeat "*" 88
echo_repeat "*" 88

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

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

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

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

print_instance_details "${instance_details}"

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

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

```

echo_repeat "*" 88
echo_repeat "*" 88

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

#####
# function clean_up
#
# This function cleans up the created resources.
# $1 - The name of the ec2 key pair to delete.
# $2 - The name of the key file to delete.
# $3 - The ID of the security group to delete.
# $4 - The ID of the instance to terminate.
# $5 - The ID of the elastic IP address to release.
# $6 - The ID of the elastic IP address to disassociate.
#
# Returns:
# 0 - If successful.
# 1 - If an error occurred.
#####
function clean_up() {
    local result=0
    local key_pair_name=$1
    local key_file_name=$2
    local security_group_id=$3
    local instance_id=$4
    local allocation_id=$5
    local association_id=$6

    if [ -n "$association_id" ]; then
        # bashsupport disable=BP2002
        if (ec2_disassociate_address -a "$association_id"); then

```

```
        echo "Disassociated elastic IP address with ID $association_id"
    else
        errecho "The elastic IP address disassociation failed."
        result=1
    fi
fi

if [ -n "$allocation_id" ]; then
    # bashsupport disable=BP2002
    if (ec2_release_address -a "$allocation_id"); then
        echo "Released elastic IP address with ID $allocation_id"
    else
        errecho "The elastic IP address release failed."
        result=1
    fi
fi

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

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

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

if [ -n "$key_pair_name" ]; then
    # bashsupport disable=BP2002
    if (ec2_delete_keypair -n "$key_pair_name"); then
        echo "Deleted key pair named $key_pair_name"
    else
```

```

        errecho "The key pair delete failed."
        result=1
    fi
fi

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

return $result
}

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

```

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
}
}
```

このシナリオで使用される DynamoDB 関数。

```
#####
# 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 getopts "n:f:h" option; do
    case "${option}" in
        n) key_pair_name="${OPTARG}" ;;
        f) file_path="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

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

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

response=$(aws ec2 create-key-pair \
    --key-name "$key_pair_name" \
    --query 'KeyMaterial' \
```

```

--output text) || {
aws_cli_error_log ${?}
errecho "ERROR: AWS reports create-access-key operation failed.$response"
return 1
}

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

return 0
}

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

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

    # Retrieve the calling parameters.
    while getopt "h" option; do
        case "${option}" in
            h)
                usage
                return 0

```

```

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

local response

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

echo "$response"

return 0
}

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

```



```
local security_group_name security_group_description response

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

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

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

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

# Create the security group
```

```

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

echo "$response"
return 0
}

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

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

  # Retrieve the calling parameters.

```

```

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

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

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

local error_code=${?}

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

echo "$response"

return 0
}

#####
# function ec2_authorize_security_group_ingress
#

```

```

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

```

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

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

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

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

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

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

response=$(aws ec2 authorize-security-group-ingress \
    --group-id "$security_group_id" \
    --cidr "${ip_address}/32" \
    --protocol "$protocol" \
    --port "$from_port-$to_port" \
```

```

        --output text) || {
        aws_cli_error_log ${?}
        errecho "ERROR: AWS reports authorize-security-group-ingress operation
failed.$response"
        return 1
    }

    return 0
}

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

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

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

```

```

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

local aws_cli_args=()

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

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

echo "$response"

return 0
}

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

```

```

# The function prints the instance type and supported architecture for each
# matching instance type.
#####
function ec2_describe_instance_types() {
    local architecture=""
    local instance_types=""

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

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

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

```



```

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

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

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

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

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

local error_code=$?

```

```

rm "$tmp_json_file"

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

echo "$response"
return 0
}

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

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

```

```
    echo "  -c count - The number of instances to launch (default: 1)."  
    echo "  -h - Display help."  
    echo ""  
}  
  
# Retrieve the calling parameters.  
while getopts "i:t:k:s:c:h" option; do  
    case "${option}" in  
        i) image_id="${OPTARG}" ;;  
        t) instance_type="${OPTARG}" ;;  
        k) key_pair_name="${OPTARG}" ;;  
        s) security_group_id="${OPTARG}" ;;  
        c) count="${OPTARG}" ;;  
        h)  
            usage  
            return 0  
            ;;  
        \?)  
            echo "Invalid parameter"  
            usage  
            return 1  
            ;;  
    esac  
done  
export OPTIND=1  
  
if [[ -z "$image_id" ]]; then  
    errecho "ERROR: You must provide an Amazon Machine Image (AMI) ID with the -i  
parameter."  
    usage  
    return 1  
fi  
  
if [[ -z "$instance_type" ]]; then  
    errecho "ERROR: You must provide an instance type with the -t parameter."  
    usage  
    return 1  
fi  
  
if [[ -z "$key_pair_name" ]]; then  
    errecho "ERROR: You must provide a key pair name with the -k parameter."  
    usage  
    return 1  
fi
```

```

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

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

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

echo "$response"

return 0
}

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

```

```
function ec2_describe_instances() {
    local instance_id query response
    local option OPTARG # Required to use 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 getopts "i:h" option; do
    case "${option}" in
        i) instance_ids="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

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

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

return 0
}

#####
# function ec2_start_instances
#
# This function starts one or more Amazon Elastic Compute Cloud (Amazon EC2)
instances.
#
# Parameters:

```

```

#       -i instance_id - The ID(s) of the instance(s) to start (comma-separated).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
#####
function ec2_start_instances() {
    local instance_ids
    local option OPTARG # Required to use 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
}

```



```

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
}

```

```

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

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

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

return 0
}

```



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

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

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

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

```

    errecho "ERROR: You must provide a security group ID with the -i parameter."
    usage
    return 1
fi

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

return 0
}

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

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

    # Retrieve the calling parameters.
    while 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
}

```

このシナリオで使われるユーティリティ関数。

```

#####
# 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
}
```

- APIの詳細については、AWS CLI コマンドリファレンスの以下のトピックを参照してください。
 - [AllocateAddress](#)
 - [AssociateAddress](#)
 - [AuthorizeSecurityGroupIngress](#)
 - [CreateKeyPair](#)
 - [CreateSecurityGroup](#)
 - [DeleteKeyPair](#)

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

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

コマンドプロンプトでシナリオを実行します。

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

EC2 アクションをラップするクラスを定義します。

```
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 "t2.2xlarge"
    * 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.getInstanceTypes();
                for (InstanceTypeInfo type : instanceTypes) {
                    logger.info("The memory information of this type is " +
                    type.getMemoryInfo().getSizeInMiB());
                    logger.info("Network information is " +
                    type.getNetworkInfo().toString());
                    logger.info("Instance type is " +
                    type.getInstanceType().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;
    });
}
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
 - [AllocateAddress](#)
 - [AssociateAddress](#)
 - [AuthorizeSecurityGroupIngress](#)
 - [CreateKeyPair](#)
 - [CreateSecurityGroup](#)
 - [DeleteKeyPair](#)
 - [DeleteSecurityGroup](#)
 - [DescribeImages](#)

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

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

このファイルには、EC2 で使用される一般的なアクションのリストが含まれています。これらの手順は、インタラクティブな例を簡単に実行できるようにするシナリオフレームワークを使って構成されています。完全なコンテキストについては、GitHub リポジトリを参照してください。

```
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[provideKeyPairName.name] }),
      );

      state.keyPairId = KeyPairId;

      // Save the private key in a temporary location.
      state.tmpDirectory = await mkdtemp(join(tmpdir(), "ec2-scenario-tmp"));
      await writeFile(
        `${state.tmpDirectory}/${state[provideKeyPairName.name]}.pem`,
        KeyMaterial,
        {
          mode: 0o400,
        },
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidKeyPair.Duplicate"
      ) {
        caught.message = `${caught.message}. Try another key name.`;
      }

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

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

```
    { skipWhen: skipWhenErrors },
  );

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

export const maybeDeleteKeyPair = new ScenarioAction(
  "deleteKeyPair",
  async (/** @type {State} */ state) => {
    try {
      // Delete a key pair by name from EC2
      await state.ec2Client.send(
        new DeleteKeyPairCommand({ KeyName: state[provideKeyPairName.name] }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        // Occurs when a required parameter (e.g. KeyName) is undefined.
        caught.name === "MissingParameter"
      ) {
        caught.message = `${caught.message}. Did you provide the required value?`;
      }
      state.errors.push(caught);
    }
  },
  {
    // Don't do anything when there's no key pair to delete or the user chooses
    // to keep it.
    skipWhen: (/** @type {State} */ state) =>
      !state.keyPairId || !state[confirmDeleteKeyPair.name],
  },
);

export const provideSecurityGroupName = new ScenarioInput(
  "securityGroupName",
  "Provide a name for a new security group.",

```

```
{ type: "input", default: "ec2-scenario-sg", skipWhen: skipWhenErrors },
);

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

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

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

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

```
);

export const maybeDeleteSecurityGroup = new ScenarioAction(
  "deleteSecurityGroup",
  async (** @type {State} */ state) => {
    try {
      // Delete the security group if the 'skipWhen' condition below is not met.
      await state.ec2Client.send(
        new DeleteSecurityGroupCommand({
          GroupId: state.securityGroupId,
        }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidGroupId.Malformed"
      ) {
        caught.message = `${caught.message}. Please provide a valid GroupId.`;
      }
      state.errors.push(caught);
    }
  },
  {
    // Don't do anything when there's no security group to delete
    // or the user chooses to keep it.
    skipWhen: (** @type {State} */ state) =>
      !state.securityGroupId || !state[confirmDeleteSecurityGroup.name],
  },
);

export const authorizeSecurityGroupIngress = new ScenarioAction(
  "authorizeSecurity",
  async (** @type {State} */ state) => {
    try {
      // Get the public IP address of the machine running this example.
      const ipAddress = await new Promise((res, rej) => {
        get("http://checkip.amazonaws.com", (response) => {
          let data = "";
          response.on("data", (chunk) => {
            data += chunk;
          });
          response.on("end", () => res(data.trim()));
        }).on("error", (err) => {
          rej(err);
        });
      });
    }
  },
);
```

```
    });
  });
  state.ipAddress = ipAddress;
  // Allow ingress from the IP address above to the security group.
  // This will allow you to SSH into the EC2 instance.
  const command = new AuthorizeSecurityGroupIngressCommand({
    GroupId: state.securityGroupId,
    IpPermissions: [
      {
        IpProtocol: "tcp",
        FromPort: 22,
        ToPort: 22,
        IpRanges: [{ CidrIp: `${ipAddress}/32` }],
      },
    ],
  });

  await state.ec2Client.send(command);
} catch (caught) {
  if (
    caught instanceof Error &&
    caught.name === "InvalidGroupId.Malformed"
  ) {
    caught.message = `${caught.message}. Please provide a valid GroupId.`;
  }

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

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

export const getImages = new ScenarioAction(
  "images",
  async (/** @type {State} */ state) => {
    const AMIs = [];
```



```
// Some AWS services publish information about common artifacts as AWS
Systems Manager (SSM)
// public parameters. For example, the Amazon Elastic Compute Cloud (Amazon
EC2)
// service publishes information about Amazon Machine Images (AMIs) as public
parameters.

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

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

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

```

    );

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

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

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

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

export const getCompatibleInstanceTypes = new ScenarioAction(
  "getCompatibleInstanceTypes",
  async (/** @type { State } */ state) => {
    // Get more details about instance types that match the architecture of
    // the provided image.
    const paginator = paginateDescribeInstanceTypes(
      { client: state.ec2Client, pageSize: 25 },
      {

```

```
    Filters: [
      {
        Name: "processor-info.supported-architecture",
        // The value selected from provideImage()
        Values: [state.image.Architecture],
      },
      // Filter for smaller, less expensive, types.
      { Name: "instance-type", Values: ["*.micro", "/*.small"] },
    ],
  },
);

const instanceTypes = [];

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

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

  state.errors.push(caught);
}

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

export const provideInstanceType = new ScenarioInput(
  "instanceType",
  "Select an instance type.",
  {
```

```

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

export const runInstance = new ScenarioAction(
  "runInstance",
  async (** @type { State } */ state) => {
    const { Instances } = await state.ec2Client.send(
      new RunInstancesCommand({
        KeyName: state[provideKeyPairName.name],
        SecurityGroupIds: [state.securityGroupId],
        ImageId: state.image.ImageId,
        InstanceType: state[provideInstanceType.name],
        // Availability Zones have capacity limitations that may impact your
        // ability to launch instances.
        // The `RunInstances` operation will only succeed if it can allocate at
        // least the `MinCount` of instances.
        // However, EC2 will attempt to launch up to the `MaxCount` of instances,
        // even if the full request cannot be satisfied.
        // If you need a specific number of instances, use `MinCount` and
        // `MaxCount` set to the same value.
        // If you want to launch up to a certain number of instances, use
        // `MaxCount` and let EC2 provision as many as possible.
        // If you require a minimum number of instances, but do not want to
        // exceed a maximum, use both `MinCount` and `MaxCount`.
        MinCount: 1,
        MaxCount: 1,
      })),
  );

state.instanceId = Instances[0].InstanceId;

try {
  // Poll `DescribeInstanceStatus` until status is "ok".
  await waitUntilInstanceStatusOk(

```

```

        {
            client: state.ec2Client,
            maxWaitTime: MAX_WAITER_TIME_IN_SECONDS,
        },
        { InstanceIds: [Instances[0].InstanceId] },
    );
} catch (caught) {
    if (caught instanceof Error && caught.name === "TimeoutError") {
        caught.message = `${caught.message}. Try increasing the maxWaitTime in
the waiter.`;
    }

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

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

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

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

            for await (const page of paginator) {
                for (const reservation of page.Reservations) {

```

```

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

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

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

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

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

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

```

```
);

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

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

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

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

export const startInstance = new ScenarioAction(
  "startInstance",
  async (/** @type { State } */ state) => {
    try {
```

```
    await state.ec2Client.send(
      new StartInstancesCommand({
        InstanceIds: [state.instanceId],
      }),
    );

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

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

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

export const allocateIp = new ScenarioAction(
  "allocateIp",
  async (** @type { State } */ state) => {
    try {
      // An Elastic IP address is allocated to your AWS account, and is yours
      until you release it.
      const { AllocationId, PublicIp } = await state.ec2Client.send(
        new AllocateAddressCommand({}),
      );
    }
  }
);
```



```

        state.allocationId = AllocationId;
        state.allocatedIpAddress = PublicIp;
    } catch (caught) {
        if (caught instanceof Error && caught.name === "MissingParameter") {
            caught.message = `${caught.message}. Did you provide these values?`;
        }
        state.errors.push(caught);
    }
},
{ skipWhen: skipWhenErrors },
);

export const associateIp = new ScenarioAction(
    "associateIp",
    async (/** @type { State } */ state) => {
        try {
            // Associate an allocated IP address to an EC2 instance. An IP address can
            // be allocated
            // with the AllocateAddress action.
            const { AssociationId } = await state.ec2Client.send(
                new AssociateAddressCommand({
                    AllocationId: state.allocationId,
                    InstanceId: state.instanceId,
                }),
            );
            state.associationId = AssociationId;
            // Update the IP address that is being tracked to match
            // the one just associated.
            state.instanceIpAddress = state.allocatedIpAddress;
        } catch (caught) {
            if (
                caught instanceof Error &&
                caught.name === "InvalidAllocationID.NotFound"
            ) {
                caught.message = `${caught.message}. Did you provide the ID of a valid
                Elastic IP address AllocationId?`;
            }
            state.errors.push(caught);
        }
    },
    { skipWhen: skipWhenErrors },
);

export const logStaticIpProof = new ScenarioOutput(

```

```
    "logStaticIpProof",
    "The IP address should remain the same even after stopping and starting the
instance.",
    { header: true, skipWhen: skipWhenErrors },
);

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

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

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

```

```
    skipWhen: (/** @type { State } */ state) =>
      !state[confirmDisassociateAddress.name] || !state.associationId,
  },
);

export const maybeReleaseAddress = new ScenarioAction(
  "maybeReleaseAddress",
  async (/** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new ReleaseAddressCommand({
          AllocationId: state.allocationId,
        }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidAllocationID.NotFound"
      ) {
        caught.message = `${caught.message}. Please provide a valid
AllocationID.`;
      }
      state.errors.push(caught);
    }
  },
  {
    skipWhen: (/** @type { State } */ state) =>
      !state[confirmDisassociateAddress.name] || !state.allocationId,
  },
);

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

export const maybeTerminateInstance = new ScenarioAction(
  "terminateInstance",
  async (/** @type { State } */ state) => {
```

```

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

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

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

export const logErrors = new ScenarioOutput(
  "logErrors",
  (/** @type {State} */ state) => {
    const errorList = state.errors
      .map((err) => ` - ${err.name}: ${err.message}`)
      .join("\n");
  },
);

```

```
    return `Scenario errors found:\n${errorList}`;  
  },  
  {  
    preformatted: true,  
    header: true,  
    // Don't log errors when there aren't any!  
    skipWhen: (** @type {State} */ state) => state.errors.length === 0,  
  },  
);
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の以下のトピックを参照してください。
 - [AllocateAddress](#)
 - [AssociateAddress](#)
 - [AuthorizeSecurityGroupIngress](#)
 - [CreateKeyPair](#)
 - [CreateSecurityGroup](#)
 - [DeleteKeyPair](#)
 - [DeleteSecurityGroup](#)
 - [DescribeImages](#)
 - [DescribeInstanceTypes](#)
 - [DescribeInstances](#)
 - [DescribeKeyPairs](#)
 - [DescribeSecurityGroups](#)
 - [DisassociateAddress](#)
 - [ReleaseAddress](#)
 - [RunInstances](#)
 - [StartInstances](#)
 - [StopInstances](#)
 - [TerminateInstances](#)
 - [UnmonitorInstances](#)

Kotlin

SDK for Kotlin

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This Kotlin example performs the following tasks:

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

val DASHES = String(CharArray(80)).replace("\u0000", "-")
```

```
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <keyName> <fileName> <groupName> <groupDesc> <vpcId> <myIpAddress>

        Where:
            keyName - A key pair name (for example, TestKeyPair).
            fileName - A file name where the key information is written to.
            groupName - The name of the security group.
            groupDesc - The description of the security group.
            vpcId - A VPC ID. You can get this value from the AWS Management
Console.
            myIpAddress - The IP address of your development machine.

        """

    if (args.size != 6) {
        println(usage)
        exitProcess(0)
    }

    val keyName = args[0]
    val fileName = args[1]
    val groupName = args[2]
    val groupDesc = args[3]
    val vpcId = args[4]
    val myIpAddress = args[5]
    var newInstanceId: String? = ""

    println(DASHES)
    println("Welcome to the Amazon EC2 example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. Create an RSA key pair and save the private key material as
a .pem file.")
    createKeyPairSc(keyName, fileName)
    println(DASHES)

    println(DASHES)
    println("2. List key pairs.")
    describeEC2KeysSc()
    println(DASHES)
}
```

```
println(DASHES)
println("3. Create a security group.")
val groupId = createEC2SecurityGroupSc(groupName, groupDesc, vpcId,
myIpAddress)
println(DASHES)

println(DASHES)
println("4. Display security group info for the newly created security
group.")
describeSecurityGroupsSc(groupId.toString())
println(DASHES)

println(DASHES)
println("5. Get a list of Amazon Linux 2 AMIs and select one with amzn2 in
the name.")
val instanceId = getParaValuesSc()
if (instanceId == "") {
    println("The instance Id value isn't valid.")
    exitProcess(0)
}
println("The instance Id is $instanceId.")
println(DASHES)

println(DASHES)
println("6. Get more information about an amzn2 image and return the AMI
value.")
val amiValue = instanceId?.let { describeImageSc(it) }
if (instanceId == "") {
    println("The instance Id value is invalid.")
    exitProcess(0)
}
println("The AMI value is $amiValue.")
println(DASHES)

println(DASHES)
println("7. Get a list of instance types.")
val instanceType = getInstanceTypesSc()
println(DASHES)

println(DASHES)
println("8. Create an instance.")
if (amiValue != null) {
    newInstanceId = runInstanceSc(instanceType, keyName, groupName, amiValue)
```



```
        println("The instance Id is $newInstanceId")
    }
    println(DASHES)

    println(DASHES)
    println("9. Display information about the running instance. ")
    var ipAddress = describeEC2InstancesSc(newInstanceId)
    println("You can SSH to the instance using this command:")
    println("ssh -i " + fileName + "ec2-user@" + ipAddress)
    println(DASHES)

    println(DASHES)
    println("10. Stop the instance.")
    if (newInstanceId != null) {
        stopInstanceSc(newInstanceId)
    }
    println(DASHES)

    println(DASHES)
    println("11. Start the instance.")
    if (newInstanceId != null) {
        startInstanceSc(newInstanceId)
    }
    ipAddress = describeEC2InstancesSc(newInstanceId)
    println("You can SSH to the instance using this command:")
    println("ssh -i " + fileName + "ec2-user@" + ipAddress)
    println(DASHES)

    println(DASHES)
    println("12. Allocate an Elastic IP address and associate it with the
instance.")
    val allocationId = allocateAddressSc()
    println("The allocation Id value is $allocationId")
    val associationId = associateAddressSc(newInstanceId, allocationId)
    println("The associate Id value is $associationId")
    println(DASHES)

    println(DASHES)
    println("13. Describe the instance again.")
    ipAddress = describeEC2InstancesSc(newInstanceId)
    println("You can SSH to the instance using this command:")
    println("ssh -i " + fileName + "ec2-user@" + ipAddress)
    println(DASHES)
```

```
println(DASHES)
println("14. Disassociate and release the Elastic IP address.")
disassociateAddressSc(associationId)
releaseEC2AddressSc(allocationId)
println(DASHES)

println(DASHES)
println("15. Terminate the instance and use a waiter.")
if (newInstanceId != null) {
    terminateEC2Sc(newInstanceId)
}
println(DASHES)

println(DASHES)
println("16. Delete the security group.")
if (groupId != null) {
    deleteEC2SecGroupSc(groupId)
}
println(DASHES)

println(DASHES)
println("17. Delete the key pair.")
deleteKeysSc(keyName)
println(DASHES)

println(DASHES)
println("You successfully completed the Amazon EC2 scenario.")
println(DASHES)
}

suspend fun deleteKeysSc(keyPair: String) {
    val request =
        DeleteKeyPairRequest {
            keyName = keyPair
        }
    Ec2Client.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")
    }
}
}

```

- APIの詳細については、「AWS SDK for Kotlin API リファレンス」の以下のトピックを参照してください。

- [AllocateAddress](#)

- [AssociateAddress](#)
- [AuthorizeSecurityGroupIngress](#)
- [CreateKeyPair](#)
- [CreateSecurityGroup](#)
- [DeleteKeyPair](#)
- [DeleteSecurityGroup](#)
- [DescribeImages](#)
- [DescribeInstanceTypes](#)
- [DescribeInstances](#)
- [DescribeKeyPairs](#)
- [DescribeSecurityGroups](#)
- [DisassociateAddress](#)
- [ReleaseAddress](#)
- [RunInstances](#)
- [StartInstances](#)
- [StopInstances](#)
- [TerminateInstances](#)
- [UnmonitorInstances](#)

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブなシナリオを実行します。

```
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.")

```

キーペアアクションをラップするクラスを定義します。

```
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

```


セキュリティグループのアクションをラップするクラスを定義します。

```
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

```

インスタンスアクションをラップするクラスを定義します。

```

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

```

Elastic IP アクションをラップするクラスを定義します。

```

class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
    actions using the client interface."""

```

```

class ElasticIp:
    """Represents an Elastic IP and its associated instance."""

    def __init__(
        self, allocation_id: str, public_ip: str, instance_id: Optional[str]
= None
    ) -> None:
        """
        Initializes the ElasticIp object.

        :param allocation_id: The allocation ID of the Elastic IP.
        :param public_ip: The public IP address of the Elastic IP.
        :param instance_id: The ID of the associated EC2 instance, if any.
        """
        self.allocation_id = allocation_id
        self.public_ip = public_ip
        self.instance_id = instance_id

    def __init__(self, ec2_client: Any) -> None:
        """
        Initializes the ElasticIpWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client
        self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def allocate(self) -> "ElasticIpWrapper.ElasticIp":
        """
        Allocates an Elastic IP address that can be associated with an Amazon EC2

```

```

        instance. By using an Elastic IP address, you can keep the public IP
        address
        constant even when you restart the associated instance.

        :return: The ElasticIp object for the newly created Elastic IP address.
        :raises ClientError: If the allocation fails, such as reaching the
        maximum limit of Elastic IPs.
        """
        try:
            response = self.ec2_client.allocate_address(Domain="vpc")
            elastic_ip = self.ElasticIp(
                allocation_id=response["AllocationId"],
                public_ip=response["PublicIp"]
            )
            self.elastic_ips.append(elastic_ip)
        except ClientError as err:
            if err.response["Error"]["Code"] == "AddressLimitExceeded":
                logger.error(
                    "Max IP's reached. Release unused addresses or contact AWS
                    Support for an increase."
                )
            raise err
        return elastic_ip

    def associate(
        self, allocation_id: str, instance_id: str
    ) -> Union[Dict[str, Any], None]:
        """
        Associates an Elastic IP address with an instance. When this association
        is
        created, the Elastic IP's public IP address is immediately used as the
        public
        IP address of the associated instance.

        :param allocation_id: The allocation ID of the Elastic IP.
        :param instance_id: The ID of the Amazon EC2 instance.
        :return: A response that contains the ID of the association, or None if
        no Elastic IP is found.
        :raises ClientError: If the association fails, such as when the instance
        ID is not found.
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
            allocation_id)

```

```

        if elastic_ip is None:
            logger.info(f"No Elastic IP found with allocation ID
{allocation_id}.")
            return None

        try:
            response = self.ec2_client.associate_address(
                AllocationId=allocation_id, InstanceId=instance_id
            )
            elastic_ip.instance_id = (
                instance_id # Track the instance associated with this Elastic
IP.
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidInstanceID.NotFound":
                logger.error(
                    f"Failed to associate Elastic IP {allocation_id} with
{instance_id} "
                    "because the specified instance ID does not exist or has not
propagated fully. "
                    "Verify the instance ID and try again, or wait a few moments
before attempting to "
                    "associate the Elastic IP address."
                )
                raise
            return response

    def disassociate(self, allocation_id: str) -> None:
        """
        Removes an association between an Elastic IP address and an instance.
When the
        association is removed, the instance is assigned a new public IP address.

        :param allocation_id: The allocation ID of the Elastic IP to
disassociate.
        :raises ClientError: If the disassociation fails, such as when the
association ID is not found.
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
        if elastic_ip is None or elastic_ip.instance_id is None:
            logger.info(

```

```

        f"No association found for Elastic IP with allocation ID
{allocation_id}."
    )
    return

    try:
        # Retrieve the association ID before disassociating
        response =
self.ec2_client.describe_addresses(AllocationIds=[allocation_id])
        association_id = response["Addresses"][0].get("AssociationId")

        if association_id:

self.ec2_client.disassociate_address(AssociationId=association_id)
            elastic_ip.instance_id = None # Remove the instance association
        else:
            logger.info(
                f"No Association ID found for Elastic IP with allocation ID
{allocation_id}."
            )

    except ClientError as err:
        if err.response["Error"]["Code"] == "InvalidAssociationID.NotFound":
            logger.error(
                f"Failed to disassociate Elastic IP {allocation_id} "
                "because the specified association ID for the Elastic IP
address was not found. "
                "Verify the association ID and ensure the Elastic IP is
currently associated with a "
                "resource before attempting to disassociate it."
            )
            raise

def release(self, allocation_id: str) -> None:
    """
    Releases an Elastic IP address. After the Elastic IP address is released,
    it can no longer be used.

    :param allocation_id: The allocation ID of the Elastic IP to release.
    :raises ClientError: If the release fails, such as when the Elastic IP
address is not found.
    """

```



```

        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
        if elastic_ip is None:
            logger.info(f"No Elastic IP found with allocation ID
{allocation_id}.")
            return

        try:
            self.ec2_client.release_address(AllocationId=allocation_id)
            self.elastic_ips.remove(elastic_ip) # Remove the Elastic IP from the
list
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidAddress.NotFound":
                logger.error(
                    f"Failed to release Elastic IP address {allocation_id} "
                    "because it could not be found. Verify the Elastic IP address
"
                    "and ensure it is allocated to your account in the correct
region "
                    "before attempting to release it."
                )
            raise

    @staticmethod
    def get_elastic_ip_by_allocation(
        elastic_ips: List["ElasticIpWrapper.ElasticIp"], allocation_id: str
    ) -> Optional["ElasticIpWrapper.ElasticIp"]:
        """
        Retrieves an Elastic IP object by its allocation ID from a given list of
Elastic IPs.

        :param elastic_ips: A list of ElasticIp objects.
        :param allocation_id: The allocation ID of the Elastic IP to retrieve.
        :return: The ElasticIp object associated with the allocation ID, or None
if not found.
        """
        return next(
            (ip for ip in elastic_ips if ip.allocation_id == allocation_id), None
        )

```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の以下のトピックを参照してください。
 - [AllocateAddress](#)
 - [AssociateAddress](#)
 - [AuthorizeSecurityGroupIngress](#)
 - [CreateKeyPair](#)
 - [CreateSecurityGroup](#)
 - [DeleteKeyPair](#)
 - [DeleteSecurityGroup](#)
 - [DescribeImages](#)
 - [DescribeInstanceTypes](#)
 - [DescribeInstances](#)
 - [DescribeKeyPairs](#)
 - [DescribeSecurityGroups](#)
 - [DisassociateAddress](#)
 - [ReleaseAddress](#)
 - [RunInstances](#)
 - [StartInstances](#)
 - [StopInstances](#)
 - [TerminateInstances](#)
 - [UnmonitorInstances](#)

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

    //! 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@{} \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 構造体はテスト用の自動モックポイントとして機能し、その関数は EC2 SDK 呼び出しをラップします。

```
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(())
}

pub 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 構造体はテスト用の自動モックポイントとして機能し、その関数は SSM SDK 呼び出しをラップします。

```

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)
    }
}
```

このシナリオでは、シナリオ全体を通じて作成および削除されるリソースへのアクセスを管理するために、複数の「Manager」スタイルの構造体が使用されています。


```

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."),
}
}
}

```

シナリオの主なエントリポイント。

```

use ec2_code_examples::{
    ec2::EC2,
    getting_started::{
        scenario::{run, Ec2InstanceScenario},
        util::UtilImpl,
    },
    ssm::SSM,
};

#[tokio::main]
async fn main() {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::load_from_env().await;
    let ec2 = EC2::new(aws_sdk_ec2::Client::new(&sdk_config));
    let ssm = SSM::new(aws_sdk_ssm::Client::new(&sdk_config));
    let util = UtilImpl {};
    let scenario = Ec2InstanceScenario::new(ec2, ssm, util);
    run(scenario).await;
}

```

- APIの詳細については、「AWS SDK for Rust API リファレンス」の以下のトピックを参照してください。
 - [AllocateAddress](#)
 - [AssociateAddress](#)
 - [AuthorizeSecurityGroupIngress](#)

- [CreateKeyPair](#)
- [CreateSecurityGroup](#)
- [DeleteKeyPair](#)
- [DeleteSecurityGroup](#)
- [DescribeImages](#)
- [DescribeInstanceTypes](#)
- [DescribeInstances](#)
- [DescribeKeyPairs](#)
- [DescribeSecurityGroups](#)
- [DisassociateAddress](#)
- [ReleaseAddress](#)
- [RunInstances](#)
- [StartInstances](#)
- [StopInstances](#)
- [TerminateInstances](#)
- [UnmonitorInstances](#)

Swift

SDK for Swift

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

Package.swift ファイル。

```
// 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 ファイル。

```

// 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)
        }
    }
}
}

```

- APIの詳細については、AWS SDK for Swift API リファレンスの以下のトピックを参照してください。
 - [AllocateAddress](#)
 - [AssociateAddress](#)
 - [AuthorizeSecurityGroupIngress](#)
 - [CreateKeyPair](#)

- [CreateSecurityGroup](#)
- [DeleteKeyPair](#)
- [DeleteSecurityGroup](#)
- [DescribeImages](#)
- [DescribeInstanceTypes](#)
- [DescribeInstances](#)
- [DescribeKeyPairs](#)
- [DescribeSecurityGroups](#)
- [DisassociateAddress](#)
- [ReleaseAddress](#)
- [RunInstances](#)
- [StartInstances](#)
- [StopInstances](#)
- [TerminateInstances](#)
- [UnmonitorInstances](#)

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

SDK を使用した Amazon EC2 のアクション AWS SDKs

次のコード例は、AWS SDKs を使用して個々の Amazon EC2 アクションを実行する方法を示しています。それぞれの例には、GitHub へのリンクがあり、そこにはコードの設定と実行に関する説明が記載されています。

これらの抜粋は Amazon EC2 API を呼び出すもので、コンテキスト内で実行する必要がある大規模なプログラムからのコードの抜粋です。アクションは [SDK を使用した Amazon EC2 のシナリオ AWS SDKs](#) のコンテキスト内で確認できます。

以下の例には、最も一般的に使用されるアクションのみ含まれています。完全なリストについては、「[Amazon Elastic Compute Cloud API リファレンス](#)」を参照してください。

例

- [CLI で AcceptVpcPeeringConnection を使用する](#)

- [AWS SDK または CLI AllocateAddress で 使用する](#)
- [CLI で AllocateHosts を使用する](#)
- [CLI で AssignPrivateIpAddresses を使用する](#)
- [AWS SDK または CLI AssociateAddress で 使用する](#)
- [CLI で AssociateDhcpOptions を使用する](#)
- [CLI で AssociateRouteTable を使用する](#)
- [CLI で AttachInternetGateway を使用する](#)
- [CLI で AttachNetworkInterface を使用する](#)
- [CLI で AttachVolume を使用する](#)
- [CLI で AttachVpnGateway を使用する](#)
- [CLI で AuthorizeSecurityGroupEgress を使用する](#)
- [AWS SDK または CLI AuthorizeSecurityGroupIngress で 使用する](#)
- [CLI で CancelCapacityReservation を使用する](#)
- [CLI で CancellImportTask を使用する](#)
- [CLI で CancelSpotFleetRequests を使用する](#)
- [CLI で CancelSpotInstanceRequests を使用する](#)
- [CLI で ConfirmProductInstance を使用する](#)
- [CLI で CopyImage を使用する](#)
- [CLI で CopySnapshot を使用する](#)
- [CLI で CreateCapacityReservation を使用する](#)
- [CLI で CreateCustomerGateway を使用する](#)
- [CLI で CreateDhcpOptions を使用する](#)
- [CLI で CreateFlowLogs を使用する](#)
- [CLI で CreateImage を使用する](#)
- [CLI で CreateInstanceExportTask を使用する](#)
- [CLI で CreateInternetGateway を使用する](#)
- [AWS SDK または CLI CreateKeyPair で 使用する](#)
- [AWS SDK または CLI CreateLaunchTemplate で 使用する](#)

- [CLI で CreateNetworkAcl を使用する](#)
- [CLI で CreateNetworkAclEntry を使用する](#)
- [CLI で CreateNetworkInterface を使用する](#)
- [CLI で CreatePlacementGroup を使用する](#)
- [CLI で CreateRoute を使用する](#)
- [AWS SDK または CLI CreateRouteTableで を使用する](#)
- [AWS SDK または CLI CreateSecurityGroupで を使用する](#)
- [CLI で CreateSnapshot を使用する](#)
- [CLI で CreateSpotDatafeedSubscription を使用する](#)
- [AWS SDK または CLI CreateSubnetで を使用する](#)
- [AWS SDK または CLI CreateTagsで を使用する](#)
- [CLI で CreateVolume を使用する](#)
- [AWS SDK または CLI CreateVpcで を使用する](#)
- [AWS SDK または CLI CreateVpcEndpointで を使用する](#)
- [CLI で CreateVpnConnection を使用する](#)
- [CLI で CreateVpnConnectionRoute を使用する](#)
- [CLI で CreateVpnGateway を使用する](#)
- [CLI で DeleteCustomerGateway を使用する](#)
- [CLI で DeleteDhcpOptions を使用する](#)
- [CLI で DeleteFlowLogs を使用する](#)
- [CLI で DeleteInternetGateway を使用する](#)
- [AWS SDK または CLI DeleteKeyPairで を使用する](#)
- [AWS SDK または CLI DeleteLaunchTemplateで を使用する](#)
- [CLI で DeleteNetworkAcl を使用する](#)
- [CLI で DeleteNetworkAclEntry を使用する](#)
- [CLI で DeleteNetworkInterface を使用する](#)
- [CLI で DeletePlacementGroup を使用する](#)
- [CLI で DeleteRoute を使用する](#)

- [CLI で DeleteRouteTable を使用する](#)
- [AWS SDK または CLI DeleteSecurityGroupで を使用する](#)
- [AWS SDK または CLI DeleteSnapshotで を使用する](#)
- [CLI で DeleteSpotDatafeedSubscription を使用する](#)
- [CLI で DeleteSubnet を使用する](#)
- [CLI で DeleteTags を使用する](#)
- [CLI で DeleteVolume を使用する](#)
- [AWS SDK または CLI DeleteVpcで を使用する](#)
- [AWS SDK または CLI DeleteVpcEndpointsで を使用する](#)
- [CLI で DeleteVpnConnection を使用する](#)
- [CLI で DeleteVpnConnectionRoute を使用する](#)
- [CLI で DeleteVpnGateway を使用する](#)
- [CLI で DeregisterImage を使用する](#)
- [CLI で DescribeAccountAttributes を使用する](#)
- [AWS SDK または CLI DescribeAddressesで を使用する](#)
- [AWS SDK または CLI DescribeAvailabilityZonesで を使用する](#)
- [CLI で DescribeBundleTasks を使用する](#)
- [CLI で DescribeCapacityReservations を使用する](#)
- [CLI で DescribeCustomerGateways を使用する](#)
- [CLI で DescribeDhcpOptions を使用する](#)
- [CLI で DescribeFlowLogs を使用する](#)
- [CLI で DescribeHostReservationOfferings を使用する](#)
- [CLI で DescribeHosts を使用する](#)
- [AWS SDK または CLI DescribeIamInstanceProfileAssociationsで を使用する](#)
- [CLI で DescribeIdFormat を使用する](#)
- [CLI で DescribeIdentityIdFormat を使用する](#)
- [CLI で DescribeImageAttribute を使用する](#)
- [AWS SDK または CLI DescribeImagesで を使用する](#)

- [CLI で DescribeImportImageTasks を使用する](#)
- [CLI で DescribeImportSnapshotTasks を使用する](#)
- [CLI で DescribeInstanceAttribute を使用する](#)
- [AWS SDK または CLI DescribeInstanceStatus で 使用する](#)
- [AWS SDK または CLI DescribeInstanceTypes で 使用する](#)
- [AWS SDK または CLI DescribeInstances で 使用する](#)
- [CLI で DescribeInternetGateways を使用する](#)
- [AWS SDK または CLI DescribeKeyPairs で 使用する](#)
- [CLI で DescribeNetworkAcls を使用する](#)
- [CLI で DescribeNetworkInterfaceAttribute を使用する](#)
- [CLI で DescribeNetworkInterfaces を使用する](#)
- [CLI で DescribePlacementGroups を使用する](#)
- [CLI で DescribePrefixLists を使用する](#)
- [AWS SDK または CLI DescribeRegions で 使用する](#)
- [AWS SDK または CLI DescribeRouteTables で 使用する](#)
- [CLI で DescribeScheduledInstanceAvailability を使用する](#)
- [CLI で DescribeScheduledInstances を使用する](#)
- [AWS SDK または CLI DescribeSecurityGroups で 使用する](#)
- [CLI で DescribeSnapshotAttribute を使用する](#)
- [AWS SDK または CLI DescribeSnapshots で 使用する](#)
- [CLI で DescribeSpotDatafeedSubscription を使用する](#)
- [CLI で DescribeSpotFleetInstances を使用する](#)
- [CLI で DescribeSpotFleetRequestHistory を使用する](#)
- [CLI で DescribeSpotFleetRequests を使用する](#)
- [CLI で DescribeSpotInstanceRequests を使用する](#)
- [CLI で DescribeSpotPriceHistory を使用する](#)
- [AWS SDK または CLI DescribeSubnets で 使用する](#)
- [CLI で DescribeTags を使用する](#)

- [CLI で DescribeVolumeAttribute を使用する](#)
- [CLI で DescribeVolumeStatus を使用する](#)
- [CLI で DescribeVolumes を使用する](#)
- [CLI で DescribeVpcAttribute を使用する](#)
- [CLI で DescribeVpcClassicLink を使用する](#)
- [CLI で DescribeVpcClassicLinkDnsSupport を使用する](#)
- [CLI で DescribeVpcEndpointServices を使用する](#)
- [CLI で DescribeVpcEndpoints を使用する](#)
- [AWS SDK または CLI DescribeVpcsで を使用する](#)
- [CLI で DescribeVpnConnections を使用する](#)
- [CLI で DescribeVpnGateways を使用する](#)
- [CLI で DetachInternetGateway を使用する](#)
- [CLI で DetachNetworkInterface を使用する](#)
- [CLI で DetachVolume を使用する](#)
- [CLI で DetachVpnGateway を使用する](#)
- [CLI で DisableVgwRoutePropagation を使用する](#)
- [CLI で DisableVpcClassicLink を使用する](#)
- [CLI で DisableVpcClassicLinkDnsSupport を使用する](#)
- [AWS SDK または CLI DisassociateAddressで を使用する](#)
- [CLI で DisassociateRouteTable を使用する](#)
- [CLI で EnableVgwRoutePropagation を使用する](#)
- [CLI で EnableVolumelo を使用する](#)
- [CLI で EnableVpcClassicLink を使用する](#)
- [CLI で EnableVpcClassicLinkDnsSupport を使用する](#)
- [CLI で GetConsoleOutput を使用する](#)
- [CLI で GetHostReservationPurchasePreview を使用する](#)
- [AWS SDK または CLI GetPasswordDataで を使用する](#)
- [CLI で ImportImage を使用する](#)

- [CLI で ImportKeyPair を使用する](#)
- [CLI で ImportSnapshot を使用する](#)
- [CLI で ModifyCapacityReservation を使用する](#)
- [CLI で ModifyHosts を使用する](#)
- [CLI で ModifyIdFormat を使用する](#)
- [CLI で ModifyImageAttribute を使用する](#)
- [CLI で ModifyInstanceAttribute を使用する](#)
- [CLI で ModifyInstanceCreditSpecification を使用する](#)
- [CLI で ModifyNetworkInterfaceAttribute を使用する](#)
- [CLI で ModifyReservedInstances を使用する](#)
- [CLI で ModifySnapshotAttribute を使用する](#)
- [CLI で ModifySpotFleetRequest を使用する](#)
- [CLI で ModifySubnetAttribute を使用する](#)
- [CLI で ModifyVolumeAttribute を使用する](#)
- [CLI で ModifyVpcAttribute を使用する](#)
- [AWS SDK または CLI MonitorInstances で使用する](#)
- [CLI で MoveAddressToVpc を使用する](#)
- [CLI で PurchaseHostReservation を使用する](#)
- [CLI で PurchaseScheduledInstances を使用する](#)
- [AWS SDK または CLI RebootInstances で使用する](#)
- [CLI で RegisterImage を使用する](#)
- [CLI で RejectVpcPeeringConnection を使用する](#)
- [AWS SDK または CLI ReleaseAddress で使用する](#)
- [CLI で ReleaseHosts を使用する](#)
- [AWS SDK または CLI ReplaceIamInstanceProfileAssociation で使用する](#)
- [CLI で ReplaceNetworkAclAssociation を使用する](#)
- [CLI で ReplaceNetworkAclEntry を使用する](#)
- [CLI で ReplaceRoute を使用する](#)
- [CLI で ReplaceRouteTableAssociation を使用する](#)

- [CLI で ReportInstanceStatus を使用する](#)
- [CLI で RequestSpotFleet を使用する](#)
- [CLI で RequestSpotInstances を使用する](#)
- [CLI で ResetImageAttribute を使用する](#)
- [CLI で ResetInstanceAttribute を使用する](#)
- [CLI で ResetNetworkInterfaceAttribute を使用する](#)
- [CLI で ResetSnapshotAttribute を使用する](#)
- [CLI で RevokeSecurityGroupEgress を使用する](#)
- [CLI で RevokeSecurityGroupIngress を使用する](#)
- [AWS SDK または CLI RunInstancesで を使用する](#)
- [CLI で RunScheduledInstances を使用する](#)
- [AWS SDK または CLI StartInstancesで を使用する](#)
- [AWS SDK または CLI StopInstancesで を使用する](#)
- [AWS SDK または CLI TerminateInstancesで を使用する](#)
- [CLI で UnassignPrivateIpAddresses を使用する](#)
- [AWS SDK または CLI UnmonitorInstancesで を使用する](#)
- [CLI で UpdateSecurityGroupRuleDescriptionsIngress を使用する](#)

CLI で **AcceptVpcPeeringConnection** を使用する

次のサンプルコードは、AcceptVpcPeeringConnection を使用方法を説明しています。

CLI

AWS CLI

VPC ピアリング接続を承諾するには

この例では、指定された VPC ピアリング接続リクエストを承諾します。

コマンド:

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

出力:

```
{
  "VpcPeeringConnection": {
    "Status": {
      "Message": "Provisioning",
      "Code": "provisioning"
    },
    "Tags": [],
    "AccepterVpcInfo": {
      "OwnerId": "444455556666",
      "VpcId": "vpc-44455566",
      "CidrBlock": "10.0.1.0/28"
    },
    "VpcPeeringConnectionId": "pcx-1a2b3c4d",
    "RequesterVpcInfo": {
      "OwnerId": "444455556666",
      "VpcId": "vpc-111abc45",
      "CidrBlock": "10.0.0.0/28"
    }
  }
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[AcceptVpcPeeringConnection](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、リクエストされた VpcPeeringConnectionId pcx-1dfad234b56ff78be を承認します。

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

出力:

```
AccepterVpcInfo      : Amazon.EC2.Model.VpcPeeringConnectionVpcInfo
ExpirationTime       : 1/1/0001 12:00:00 AM
RequesterVpcInfo     : Amazon.EC2.Model.VpcPeeringConnectionVpcInfo
Status               : Amazon.EC2.Model.VpcPeeringConnectionStateReason
```

```
Tags                : {}  
VpcPeeringConnectionId : pcx-1dfad234b56ff78be
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[AcceptVpcPeeringConnection](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、リクエストされた VpcPeeringConnectionId pcx-1dfad234b56ff78be を承認します。

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

出力:

```
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
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[AcceptVpcPeeringConnection](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **AllocateAddress**で を使用する

次のサンプルコードは、AllocateAddress を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)
- [プライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成する](#)
- [Amazon VPC の使用を開始する](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

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

        throw;
    }
    catch (Exception ex)
```

```

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

```

- API の詳細については、AWS SDK for .NET API リファレンスの「[AllocateAddress](#)」を参照してください。

Bash

AWS CLI Bash スクリプトを使用する

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

#####
# 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
}

```

この例で使用されているユーティリティ関数。

```

#####
# 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
}

```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[AllocateAddress](#)」を参照してください。

C++

SDK for C++

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

//! Allocate an Elastic IP address and associate it with an Amazon Elastic
    Compute Cloud
//! (Amazon EC2) instance.
/*!
    \param instanceID: An EC2 instance ID.
    \param[out] publicIPAddress: String to return the public IP address.
    \param[out] allocationID: String to return the allocation ID.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::EC2::allocateAndAssociateAddress(const Aws::String &instanceId,
    Aws::String &publicIPAddress,
    Aws::String &allocationID,

```

```

const
Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::AllocateAddressRequest request;
    request.SetDomain(Aws::EC2::Model::DomainType::vpc);

    const Aws::EC2::Model::AllocateAddressOutcome outcome =
        ec2Client.AllocateAddress(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to allocate Elastic IP address:" <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }
    const Aws::EC2::Model::AllocateAddressResponse &response =
outcome.GetResult();
    allocationID = response.GetAllocationId();
    publicIPAddress = response.GetPublicIp();

    return true;
}

```

- API の詳細については、AWS SDK for C++ API リファレンスの「[AllocateAddress](#)」を参照してください。

CLI

AWS CLI

例 1: Amazon のアドレスプールから Elastic IP アドレスを割り当てるには

次の `allocate-address` の例では、Elastic IP アドレスを割り当てています。Amazon EC2 は、Amazon のアドレスプールからアドレスを選択します。

```
aws ec2 allocate-address
```

出力:

```

{
    "PublicIp": "70.224.234.241",

```

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

詳細については、「Amazon EC2 ユーザーガイド」の「[Elastic IP アドレス](#)」を参照してください。

例 2: Elastic IP アドレスを割り当て、インスタンスまたはネットワークボーダーグループと関連付けるには

次の `allocate-address` の例では、Elastic IP アドレスを割り当て、指定されたネットワークボーダーグループに関連付けます。

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

出力:

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

詳細については、「Amazon EC2 ユーザーガイド」の「[Elastic IP アドレス](#)」を参照してください。

例 3: 所有するアドレスプールから Elastic IP アドレスを割り当てるには

次の `allocate-address` の例では、Amazon Web Services アカウントに入れたアドレスプールから Elastic IP アドレスを割り当てています。Amazon EC2 は、アドレスプールからアドレスを選択します。

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

出力:

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

詳細については、「Amazon EC2 ユーザーガイド」の「[Elastic IP アドレス](#)」を参照してください。

例 4: IPAM プールから Elastic IP アドレスを割り当てるには

次の `allocate-address` の例では、Amazon VPC IP Address Manager (IPAM) プールから特定の /32 Elastic IP アドレスを割り当てます。

```
aws ec2 allocate-address \
  --region us-east-1 \
  --ipam-pool-id ipam-pool-1234567890abcdef0 \
  --address 192.0.2.0
```

出力:

```
{
  "PublicIp": "192.0.2.0",
  "AllocationId": "eipalloc-abcdef01234567890",
  "PublicIpv4Pool": "ipam-pool-1234567890abcdef0",
  "NetworkBorderGroup": "us-east-1",
  "Domain": "vpc"
}
```

詳細については「Amazon VPC IPAM ユーザーガイド」の「[Allocate sequential Elastic IP addresses from an IPAM pool](#)」を参照してください。

- API の詳細については、「AWS CLI Command Reference」の「[AllocateAddress](#)」を参照してください。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

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

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[AllocateAddress](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

/**
 * Allocates an Elastic IP address to your AWS account.
 */
export const main = async () => {
  const client = new EC2Client({});
  const command = new AllocateAddressCommand({});

  try {
    const { AllocationId, PublicIp } = await client.send(command);
    console.log("A new IP address has been allocated to your account:");
    console.log(`ID: ${AllocationId} Public IP: ${PublicIp}`);
    console.log(
      "You can view your IP addresses in the AWS Management Console for Amazon EC2. Look under Network & Security > Elastic IPs",
    );
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide these values?`);
    } else {
      throw caught;
    }
  }
};

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

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[AllocateAddress](#)」を参照してください。

Kotlin

SDK for Kotlin

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[AllocateAddress](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、VPC 内のインスタンスで使用する Elastic IP アドレスを割り当てます。

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

出力:

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

例 2: この例では、EC2-Classic のインスタンスで使用する Elastic IP アドレスを割り当てます。

```
New-EC2Address
```

出力:

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[AllocateAddress](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、VPC 内のインスタンスで使用する Elastic IP アドレスを割り当てます。

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

出力:

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

例 2: この例では、EC2-Classic のインスタンスで使用する Elastic IP アドレスを割り当てます。

```
New-EC2Address
```

出力:

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[AllocateAddress](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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:
            """
            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
address
        instance. By using an Elastic IP address, you can keep the public IP
        constant even when you restart the associated instance.

        :return: The ElasticIp object for the newly created Elastic IP address.
        :raises ClientError: If the allocation fails, such as reaching the
maximum limit of Elastic IPs.
        """
        try:
            response = self.ec2_client.allocate_address(Domain="vpc")
            elastic_ip = self.ElasticIp(

```

```

        allocation_id=response["AllocationId"],
        public_ip=response["PublicIp"]
    )
    self.elastic_ips.append(elastic_ip)
except ClientError as err:
    if err.response["Error"]["Code"] == "AddressLimitExceeded":
        logger.error(
            "Max IP's reached. Release unused addresses or contact AWS
Support for an increase."
        )
    raise err
return elastic_ip

```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[AllocateAddress](#)」を参照してください。

Ruby

SDK for Ruby

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

# Creates an Elastic IP address in Amazon Virtual Private Cloud (Amazon VPC).
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @return [String] The allocation ID corresponding to the Elastic IP address.
# @example
#   puts allocate_elastic_ip_address(Aws::EC2::Client.new(region: 'us-west-2'))
def allocate_elastic_ip_address(ec2_client)
  response = ec2_client.allocate_address(domain: 'vpc')
  response.allocation_id
rescue StandardError => e
  puts "Error allocating Elastic IP address: #{e.message}"
  'Error'
end

```

- API の詳細については、AWS SDK for Ruby API リファレンスの「[AllocateAddress](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn allocate_ip_address(&self) -> Result<AllocateAddressOutput,
EC2Error> {
    self.client
        .allocate_address()
        .domain(DomainType::Vpc)
        .send()
        .await
        .map_err(EC2Error::from)
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[AllocateAddress](#)」を参照してください。

SAP ABAP

SDK for SAP ABAP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。


```

TRY.
    oo_result = lo_ec2->allocateaddress( iv_domain = 'vpc' ).    " oo_result
is returned for testing purposes. "
    MESSAGE 'Allocated an Elastic IP address.' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.

```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「[AllocateAddress](#)」を参照してください。

Swift

SDK for Swift

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

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
    }
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[AllocateAddress](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **AllocateHosts** を使用する

次のサンプルコードは、AllocateHosts を使用する方法を説明しています。

CLI

AWS CLI

例 1: 専有ホストを割り当てるには

次の allocate-hosts の例では、eu-west-1a アベイラビリティーゾーンに 1 つの専有ホストを割り当てます。このゾーンでは m5.large インスタンスを起動できます。デフォルトでは、専有ホストはターゲットインスタンスの起動のみを承諾し、ホストリカバリはサポートしていません。

```
aws ec2 allocate-hosts \
  --instance-type m5.large \
  --availability-zone eu-west-1a \
  --quantity 1
```

出力:

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

```
]
}
```

例 2: 自動配置とホストリカバリが有効になっている専有ホストを割り当てるには

次の `allocate-hosts` の例では、自動配置とホストリカバリが有効になっている `eu-west-1a` アベイラビリティーゾーンに 1 つの専有ホストを割り当てます。

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

出力:

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

例 3: タグのある専有ホストを割り当てるには

次の `allocate-hosts` の例では、1 つの専有ホストを割り当てて、`purpose` という名前のキーと `production` の値を持つタグを適用します。

```
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}'
```

出力:

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

```
]
}
```

詳細については、「Amazon EC2 ユーザーガイド」の「[専有ホストを割り当てる](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[AllocateHosts](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたインスタンスタイプとアベイラビリティゾーンのアカウントに専有ホストを割り当てます。

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

出力:

```
h-01e23f4cd567890f3
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[AllocateHosts](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたインスタンスタイプとアベイラビリティゾーンのアカウントに専有ホストを割り当てます。

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

出力:

```
h-01e23f4cd567890f3
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[AllocateHosts](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で `AssignPrivateIpAddresses` を使用する

次のサンプルコードは、`AssignPrivateIpAddresses` を使用方法を説明しています。

CLI

AWS CLI

セカンダリプライベート IP アドレスをネットワークインターフェイスに割り当てるには

この例では、指定されたセカンダリプライベート IP アドレスを指定されたネットワークインターフェイスに割り当てます。コマンドが成功した場合、出力は返りません。

コマンド:

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

Amazon EC2 が選択したセカンダリプライベート IP アドレスをネットワークインターフェイスに割り当てるには

この例では、2 つのセカンダリプライベート IP アドレスを指定されたネットワークインターフェイスに割り当てます。Amazon EC2 は、ネットワークインターフェイスが関連付けられているサブネットの CIDR ブロック範囲内にある使用可能な IP アドレスから、これらの IP アドレスを自動的に割り当てます。コマンドが成功した場合、出力は返りません。

コマンド:

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

- API の詳細については、「AWS CLI コマンドリファレンス」の「[AssignPrivateIpAddresses](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたセカンダリプライベート IP アドレスを指定されたネットワークインターフェイスに割り当てます。

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

例 2: この例では、2 つのセカンダリプライベート IP アドレスを作成し、指定されたネットワークインターフェイスに割り当てます。

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[AssignPrivateIpAddresses](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたセカンダリプライベート IP アドレスを指定されたネットワークインターフェイスに割り当てます。

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

例 2: この例では、2 つのセカンダリプライベート IP アドレスを作成し、指定されたネットワークインターフェイスに割り当てます。

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[AssignPrivateIpAddresses](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **AssociateAddress**で を使用する

次のサンプルコードは、AssociateAddress を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

```

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

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

```

- API の詳細については、AWS SDK for .NET API リファレンスの「[AssociateAddress](#)」を参照してください。

Bash

AWS CLI Bash スクリプトを使用する

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

#####
# 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
}

```

```

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
}

```

この例で使用されているユーティリティ関数。

```

#####
# 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
}
```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[AssociateAddress](#)」を参照してください。

C++

SDK for C++

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

//! Associate an Elastic IP address with an EC2 instance.
/*!
 \param instanceId: An EC2 instance ID.
 \param allocationId: An Elastic IP allocation ID.
 \param[out] associationId: String to receive the association ID.
 \param clientConfiguration: AWS client configuration.
 \return bool: True if the address was associated with the instance; otherwise,
 false.
 */
bool AwsDoc::EC2::associateAddress(const Aws::String &instanceId, const
    Aws::String &allocationId,
                                   Aws::String &associationId,
                                   const Aws::Client::ClientConfiguration
    &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

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

    Aws::EC2::Model::AssociateAddressOutcome outcome =
    ec2Client.AssociateAddress(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to associate address " << allocationId <<
            " with instance " << instanceId << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully associated address " << allocationId <<
            " with instance " << instanceId << std::endl;
    }
}
```

```
        associationID = outcome.GetResult().GetAssociationId();
    }

    return outcome.IsSuccess();
}
```

- API の詳細については、AWS SDK for C++ API リファレンスの「[AssociateAddress](#)」を参照してください。

CLI

AWS CLI

例 1: Elastic IP アドレスをインスタンスに関連付けるには

次の `associate-address` の例では、Elastic IP アドレスを、指定した EC2 インスタンスに関連付けます。

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

出力:

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

例 2: Elastic IP アドレスをネットワークインターフェイスに関連付けるには

次の `associate-address` の例では、指定した Elastic IP アドレスを、指定したネットワークインターフェイスに関連付けます。

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

出力:

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

例 3: Elastic IP アドレスをプライベート IP アドレスに関連付けるには

次の `associate-address` の例では、指定した Elastic IP アドレスを、指定したネットワークインターフェイス内の指定したプライベート IP アドレスに関連付けます。

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

出力:

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

詳細については、「Amazon EC2 ユーザーガイド」の「[Elastic IP アドレス](#)」を参照してください。

- API の詳細については、「AWS CLI Command Reference」の「[AssociateAddress](#)」を参照してください。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/**
 * 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);
        }
    });
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[AssociateAddress](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

/**
 * Associates an Elastic IP address, or carrier IP address (for instances that
 * are in subnets in Wavelength Zones)
 * with an instance or a network interface.
 * @param {{ instanceId: string, allocationId: string }} options
 */
export const main = async ({ instanceId, allocationId }) => {
  const client = new EC2Client({});
  const command = new AssociateAddressCommand({
    // You need to allocate an Elastic IP address before associating it with an
    // instance.
    // You can do that with the AllocateAddressCommand.
    AllocationId: allocationId,
    // You need to create an EC2 instance before an IP address can be associated
    // with it.
    // You can do that with the RunInstancesCommand.
    InstanceId: instanceId,
  });

  try {
    const { AssociationId } = await client.send(command);
    console.log(
      `Address with allocation ID ${allocationId} is now associated with instance ${instanceId}.`,
      `The association ID is ${AssociationId}.`,
    );
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidAllocationID.NotFound"
    ) {

```



```
    ) {  
        console.warn(  
            `${caught.message}. Did you provide the ID of a valid Elastic IP address  
AllocationId?`,  
        );  
    } else {  
        throw caught;  
    }  
}  
};
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[AssociateAddress](#)」を参照してください。

Kotlin

SDK for Kotlin

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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  
    }  
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[AssociateAddress](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定された Elastic IP アドレスを VPC 内の指定されたインスタンスに関連付けます。

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

出力:

```
eipassoc-12345678
```

例 2: この例では、指定された Elastic IP アドレスを EC2-Classic の指定されたインスタンスに関連付けます。

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[AssociateAddress](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定された Elastic IP アドレスを VPC 内の指定されたインスタンスに関連付けます。

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

出力:

```
eipassoc-12345678
```

例 2: この例では、指定された Elastic IP アドレスを EC2-Classic の指定されたインスタンスに関連付けます。

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[AssociateAddress](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str]
            = None
        ) -> None:
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
            :param public_ip: The public IP address of the Elastic IP.
            :param instance_id: The ID of the associated EC2 instance, if any.
            """
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

        def __init__(self, ec2_client: Any) -> None:
            """
            Initializes the ElasticIpWrapper with an EC2 client.

            :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
            level
```

```

        access to AWS EC2 services.

        """
        self.ec2_client = ec2_client
        self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def associate(
        self, allocation_id: str, instance_id: str
    ) -> Union[Dict[str, Any], None]:
        """
        Associates an Elastic IP address with an instance. When this association
is
        created, the Elastic IP's public IP address is immediately used as the
public
        IP address of the associated instance.

        :param allocation_id: The allocation ID of the Elastic IP.
        :param instance_id: The ID of the Amazon EC2 instance.
        :return: A response that contains the ID of the association, or None if
no Elastic IP is found.
        :raises ClientError: If the association fails, such as when the instance
ID is not found.
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
        if elastic_ip is None:
            logger.info(f"No Elastic IP found with allocation ID
{allocation_id}.")
            return None

        try:
            response = self.ec2_client.associate_address(
                AllocationId=allocation_id, InstanceId=instance_id

```

```

    )
    elastic_ip.instance_id = (
        instance_id # Track the instance associated with this Elastic
IP.
    )
except ClientError as err:
    if err.response["Error"]["Code"] == "InvalidInstanceID.NotFound":
        logger.error(
            f"Failed to associate Elastic IP {allocation_id} with
{instance_id} "
            "because the specified instance ID does not exist or has not
propagated fully. "
            "Verify the instance ID and try again, or wait a few moments
before attempting to "
            "associate the Elastic IP address."
        )
    raise
return response

```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[AssociateAddress](#)」を参照してください。

Ruby

SDK for Ruby

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

# Associates an Elastic IP address with an Amazon Elastic Compute Cloud
# (Amazon EC2) instance.
#
# Prerequisites:
#
# - The allocation ID corresponding to the Elastic IP address.
# - The Amazon EC2 instance.

```

```
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param allocation_id [String] The ID of the allocation corresponding to
#   the Elastic IP address.
# @param instance_id [String] The ID of the instance.
# @return [String] The association ID corresponding to the association of the
#   Elastic IP address to the instance.
# @example
#   puts allocate_elastic_ip_address(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'eipalloc-04452e528a66279EX',
#     'i-033c48ef067af3dEX')
def associate_elastic_ip_address_with_instance(
  ec2_client,
  allocation_id,
  instance_id
)
  response = ec2_client.associate_address(
    allocation_id: allocation_id,
    instance_id: instance_id
  )
  response.association_id
rescue StandardError => e
  puts "Error associating Elastic IP address with instance: #{e.message}"
  'Error'
end
```

- APIの詳細については、AWS SDK for Ruby API リファレンスの「[AssociateAddress](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn associate_ip_address(
    &self,
    allocation_id: &str,
    instance_id: &str,
) -> Result<AssociateAddressOutput, EC2Error> {
    let response = self
        .client
        .associate_address()
        .allocation_id(allocation_id)
        .instance_id(instance_id)
        .send()
        .await?;
    Ok(response)
}
```

- APIの詳細については、「AWS SDK for Rust API リファレンス」の「[AssociateAddress](#)」を参照してください。

SAP ABAP

SDK for SAP ABAP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
TRY.
    oo_result = lo_ec2->associateaddress(                                " oo_result
is returned for testing purposes. "
        iv_allocationid = iv_allocation_id
        iv_instanceid = iv_instance_id ).
    MESSAGE 'Associated an Elastic IP address with an EC2 instance.' TYPE
'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- APIの詳細については、「AWS SDK for SAP ABAP API リファレンス」の「[AssociateAddress](#)」を参照してください。

Swift

SDK for Swift

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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
    }
}
```



```
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[AssociateAddress](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **AssociateDhcpOptions** を使用する

次のサンプルコードは、AssociateDhcpOptions を使用する方法を説明しています。

CLI

AWS CLI

DHCP オプションセットを VPC に関連付けるには

この例では、指定された DHCP オプションセットを指定された VPC に関連付けます。コマンドが成功した場合、出力は返りません。

コマンド:

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

デフォルトの DHCP オプションセットを VPC に関連付けるには

この例では、デフォルトの DHCP オプションセットを指定された VPC に関連付けます。コマンドが成功した場合、出力は返りません。

コマンド:

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

- API の詳細については、「AWS CLI コマンドリファレンス」の「[AssociateDhcpOptions](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定された DHCP オプションセットを指定された VPC に関連付けます。

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

例 2: この例では、デフォルトの DHCP オプションセットを指定された VPC に関連付けます。

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[AssociateDhcpOptions](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定された DHCP オプションセットを指定された VPC に関連付けます。

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

例 2: この例では、デフォルトの DHCP オプションセットを指定された VPC に関連付けます。

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[AssociateDhcpOptions](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **AssociateRouteTable** を使用する

次のサンプルコードは、AssociateRouteTable を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [プライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成する](#)
- [Amazon VPC の使用を開始する](#)

CLI

AWS CLI

ルートテーブルをサブネットに関連付けるには

この例では、指定されたルートテーブルを指定されたサブネットに関連付けます。

コマンド:

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

出力:

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

- API の詳細については、「AWS CLI コマンドリファレンス」の「[AssociateRouteTable](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたルートテーブルを指定されたサブネットに関連付けます。

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

出力:

```
rtbassoc-12345678
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[AssociateRouteTable](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたルートテーブルを指定されたサブネットに関連付けます。

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

出力:

```
rtbassoc-12345678
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[AssociateRouteTable](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **AttachInternetGateway** を使用する

次のサンプルコードは、AttachInternetGateway を使用方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [プライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成する](#)
- [Amazon VPC の使用を開始する](#)

CLI

AWS CLI

VPC にインターネットゲートウェイをアタッチするには

次の attach-internet-gateway の例では、指定したインターネットゲートウェイを特定の VPC にアタッチします。

```
aws ec2 attach-internet-gateway \  
  --internet-gateway-id igw-0d0fb496b3EXAMPLE \  
  --vpc-id vpc-01234567890123456
```

```
--vpc-id vpc-0a60eb65b4EXAMPLE
```

このコマンドでは何も出力されません。

詳細については、Amazon VPC ユーザーガイドの「[インターネットゲートウェイ](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[AttachInternetGateway](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたインターネットゲートウェイを特定の VPC にアタッチします。

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

例 2: この例では、VPC とインターネットゲートウェイを作成し、インターネットゲートウェイを VPC にアタッチします。

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[AttachInternetGateway](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたインターネットゲートウェイを特定の VPC にアタッチします。

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

例 2: この例では、VPC とインターネットゲートウェイを作成し、インターネットゲートウェイを VPC にアタッチします。

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[AttachInternetGateway](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **AttachNetworkInterface** を使用する

次のサンプルコードは、AttachNetworkInterface を使用する方法を説明しています。

CLI

AWS CLI

例 1: ネットワークインターフェイスをインスタンスにアタッチするには

次の attach-network-interface の例では、指定したネットワークインターフェイスを指定したインスタンスにアタッチします。

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

出力:

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

詳細については、「Amazon EC2 ユーザーガイド」の「[Elastic Network Interface](#)」を参照してください。

例 2: ネットワークインターフェイスを複数のネットワークを持つインスタンスにアタッチするには

次の attach-network-interface の例では、指定したネットワークインターフェイスを指定したインスタンスとネットワークカードにアタッチします。

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

```
--device-index 1
```

出力:

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

詳細については、「Amazon EC2 ユーザーガイド」の「[Elastic Network Interface](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[AttachNetworkInterface](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定したネットワークインターフェイスを指定したインスタンスにアタッチします。

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

出力:

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[AttachNetworkInterface](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定したネットワークインターフェイスを指定したインスタンスにアタッチします。

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

出力:

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の [AttachNetworkInterface](#) を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください [AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **AttachVolume** を使用する

次のサンプルコードは、AttachVolume を使用する方法を説明しています。

CLI

AWS CLI

ボリュームをインスタンスにアタッチするには

この例では、ボリューム (vol-1234567890abcdef0) を /dev/sdf としてインスタンス (i-01474ef662b89480) にアタッチします。

コマンド:

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

出力:

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

- API の詳細については、「AWS CLI コマンドリファレンス」の「[AttachVolume](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたボリュームを指定されたインスタンスにアタッチし、指定されたデバイス名で公開します。

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

出力:

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[AttachVolume](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたボリュームを指定されたインスタンスにアタッチし、指定されたデバイス名で公開します。

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

出力:

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[AttachVolume](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **AttachVpnGateway** を使用する

次のサンプルコードは、AttachVpnGateway を使用する方法を説明しています。

CLI

AWS CLI

仮想プライベートゲートウェイを VPC にアタッチするには

次の attach-vpn-gateway の例では、指定した仮想ゲートウェイを指定した VPC にアタッチします。

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

出力:

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

- API の詳細については、「AWS CLI コマンドリファレンス」の「[AttachVpnGateway](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定した仮想プライベートゲートウェイを指定した VPC にアタッチします。

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

出力:

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[AttachVpnGateway](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定した仮想プライベートゲートウェイを指定した VPC にアタッチします。

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

出力:

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[AttachVpnGateway](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **AuthorizeSecurityGroupEgress** を使用する

次のサンプルコードは、AuthorizeSecurityGroupEgress を使用する方法を説明しています。

CLI

AWS CLI

例 1: 特定のアドレス範囲へのアウトバウンドトラフィックを許可するルールを追加するには

次の authorize-security-group-egress の例では、TCP ポート 80 で、指定したアドレス範囲へのアクセスを許可するルールを追加します。

```
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}]'
```

出力:

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

例 2: 特定のセキュリティグループへのアウトバウンドトラフィックを許可するルールを追加するには

次の `authorize-security-group-egress` の例では、TCP ポート 80 で、指定したセキュリティグループへのアクセスを許可するルールを追加します。

```
aws ec2 authorize-security-group-egress \  
  --group-id sg-1234567890abcdef0 \  
  --ip-permissions  
  'IpProtocol=tcp,FromPort=80,ToPort=80,UserIdGroupPairs=[{GroupId=sg-0aad1c26bbeec5c22}]'
```

出力:

```
{  
  "Return": true,  
  "SecurityGroupRules": [  
    {  
      "SecurityGroupRuleId": "sgr-0b5dd815afcea9cc3",  
      "GroupId": "sg-1234567890abcdef0",  
      "GroupOwnerId": "123456789012",  
      "IsEgress": true,  
      "IpProtocol": "tcp",  
      "FromPort": 80,  
      "ToPort": 80,  
      "UserIdGroupPairs": [{  
        "GroupId": "sg-0aad1c26bbeec5c22"  
      }]  
    }  
  ]  
}
```

```

        "GroupOwnerId": "123456789012",
        "IsEgress": true,
        "IpProtocol": "tcp",
        "FromPort": 80,
        "ToPort": 80,
        "ReferencedGroupInfo": {
            "GroupId": "sg-0aad1c26bbeec5c22",
            "UserId": "123456789012"
        }
    }
]
}

```

詳細については、Amazon VPC ユーザーガイドの「[セキュリティグループ](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[AuthorizeSecurityGroupEgress](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、EC2-VPC の指定されたセキュリティグループの退出ルールを定義します。このルールは、TCP ポート 80 で、指定された IP アドレス範囲へのアクセスを許可します。この例で使用される構文には、PowerShell バージョン 3 以降が必要です。

```

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

```

例 2: PowerShell バージョン 2 では、New-Object を使用して IpPermission オブジェクトを作成する必要があります。

```

$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

```

例 3: この例では、TCP ポート 80 で指定されたソースセキュリティグループへのアクセスを許可します。

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"

Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission
@(@{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; UserIdGroupPairs=$ug } )
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[AuthorizeSecurityGroupEgress](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、EC2-VPC の指定されたセキュリティグループの退出ルールを定義します。このルールは、TCP ポート 80 で、指定された IP アドレス範囲へのアクセスを許可します。この例で使用される構文には、PowerShell バージョン 3 以降が必要です。

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

例 2: PowerShell バージョン 2 では、New-Object を使用して IpPermission オブジェクトを作成する必要があります。

```
$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
```

例 3: この例では、TCP ポート 80 で指定されたソースセキュリティグループへのアクセスを許可します。

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"
```

```
Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission
@( @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; UserIdGroupPairs=$ug } )
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[AuthorizeSecurityGroupEgress](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **AuthorizeSecurityGroupIngress**で 使用する

次のサンプルコードは、AuthorizeSecurityGroupIngress を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)
- [プライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成する](#)
- [Amazon VPC の使用を開始する](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Authorize the local computer ingress to EC2 instances associated
/// with the virtual private cloud (VPC) security group.
/// </summary>
/// <param name="groupName">The name of the security group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> AuthorizeSecurityGroupIngress(string groupName)
{
```

```

    try
    {
        // Get the IP address for the local computer.
        var ipAddress = await GetIpAddress();
        Console.WriteLine($"Your IP address is: {ipAddress}");
        var ipRanges =
            new List<IpRange> { new IpRange { CidrIp = $"{ipAddress}/32" } };
        var permission = new IpPermission
        {
            Ipv4Ranges = ipRanges,
            IpProtocol = "tcp",
            FromPort = 22,
            ToPort = 22
        };
        var permissions = new List<IpPermission> { permission };
        var response = await _amazonEC2.AuthorizeSecurityGroupIngressAsync(
            new AuthorizeSecurityGroupIngressRequest(groupName,
permissions));
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidPermission.Duplicate")
        {
            _logger.LogError(
                $"The ingress rule already exists. {ec2Exception.Message}");
        }

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

/// <summary>
/// Authorize the local computer for ingress to
/// the Amazon EC2 SecurityGroup.
/// </summary>
/// <returns>The IPv4 address of the computer running the scenario.</returns>
private static async Task<string> GetIpAddress()

```



```
{
    var httpClient = new HttpClient();
    var ipString = await httpClient.GetStringAsync("https://
checkip.amazonaws.com");

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

- API の詳細については、AWS SDK for .NET API リファレンスの「[AuthorizeSecurityGroupIngress](#)」を参照してください。

Bash

AWS CLI Bash スクリプトを使用する

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
#####
# 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 getopt "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
}
```

この例で使用されているユーティリティ関数。

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

- API の詳細については、「AWS CLI コマンドリファレンス」の「[AuthorizeSecurityGroupIngress](#)」を参照してください。

C++

SDK for C++

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

//! 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();
}

```

進入ルールを構築するユーティリティ関数。

```

//! 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);
}

```

- APIの詳細については、AWS SDK for C++ API リファレンスの「[AuthorizeSecurityGroupIngress](#)」を参照してください。

CLI

AWS CLI

例 1: インバウンド SSH トラフィックを許可するルールを追加するには

次の `authorize-security-group-ingress` の例では、TCP ポート 22 (SSH) にインバウンドトラフィックを許可するルールを追加します。

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

出力:

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

例 2: セキュリティグループからの HTTP トラフィックを許可するルールを追加するには

次の `authorize-security-group-ingress` の例では、ソースセキュリティグループ `sg-1a2b3c4d` からの TCP ポート 80 へのインバウンドアクセスを許可するルールを追加します。ソースグループは、同じ VPC にあるか、ピア VPC (VPC ピアリング接続が必要) に存在している必要があります。着信トラフィックは、ソースセキュリティグループに関連付けられたインスタンスのプライベート IP アドレスに基づいて許可されます (パブリック IP アドレスまたは Elastic IP アドレスは考慮されません)。

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

出力:

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

例 3: 同じ呼び出しに複数のルールを追加するには

次の `authorize-security-group-ingress` の例では、`ip-permissions` パラメータを使用して 2 つのインバウンドルールを追加します。一方は TCP ポート 3389 (RDP) でのインバウンドアクセスを有効にするルールであり、もう一方は ping/ICMP を有効にするルールです。

```
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}]'
```

出力:


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

例 4: ICMP トラフィックのルールを追加するには

次の `authorize-security-group-ingress` の例では、`ip-permissions` パラメータを使用して、どこからでも ICMP メッセージ `Destination Unreachable: Fragmentation Needed and Don't Fragment was Set` (タイプ 3、コード 4) を許可するインバウンドルールを追加します。

```
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}]'
```

出力:

```
{
  "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"
      }
    ]
  }
}

```

例 5: IPv6 トラフィックのルールを追加するには

次の `authorize-security-group-ingress` の例では、`ip-permissions` パラメータを使用して、IPv6 範囲 `2001:db8:1234:1a00::/64` からの SSH アクセス (ポート 22) を許可するインバウンドルールを追加します。

```

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}]'

```

出力:

```

{
  "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"
    }
  ]
}

```

例 6: ICMPv6 トラフィックのルールを追加するには

次の `authorize-security-group-ingress` の例では、`ip-permissions` パラメータを使用して、どこからでも ICMPv6 トラフィックを許可するインバウンドルールを追加します。

```
aws ec2 authorize-security-group-ingress \
  --group-id sg-1234567890abcdef0 \
  --ip-permissions 'IpProtocol=icmpv6,Ipv6Ranges=[{CidrIpv6=::/0}]'
```

出力:

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

例 7: 説明付きのルールを追加する

次の `authorize-security-group-ingress` の例では、`ip-permissions` パラメータを使用して、指定した IPv4 アドレス範囲からの RDP トラフィックを許可するインバウンドルールを追加します。ルールには、後で特定できるように説明が含まれます。

```
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'}]'
```

出力:

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

例 8: プレフィックスリストを使用するインバウンドルールを追加するには

次の `authorize-security-group-ingress` の例では、`ip-permissions` パラメータを使用して、指定されたプレフィックスリスト内の CIDR 範囲のトラフィックすべてを許可するインバウンドルールを追加します。

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

出力:

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

```
    }
  ]
}
```

詳細については、Amazon VPC ユーザーガイドの「[セキュリティグループ](#)」を参照してください。

- API の詳細については、「AWS CLI Command Reference」の「[AuthorizeSecurityGroupIngress](#)」を参照してください。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

```
.groupName(groupName)
.description(groupDesc)
.vpcId(vpcId)
.build();

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

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

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

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

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

```

        }
    }
});
}

```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[AuthorizeSecurityGroupIngress](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

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

/**
 * Adds the specified inbound (ingress) rules to a security group.
 * @param {{ groupId: string, ipAddress: string }} options
 */
export const main = async ({ groupId, ipAddress }) => {
  const client = new EC2Client({});
  const command = new AuthorizeSecurityGroupIngressCommand({
    // Use a group ID from the AWS console or
    // the DescribeSecurityGroupsCommand.
    GroupId: groupId,
    IpPermissions: [
      {
        IpProtocol: "tcp",
        FromPort: 22,
        ToPort: 22,
        // The IP address to authorize.
        // For more information on this notation, see

```

```
// https://en.wikipedia.org/wiki/Classless_Inter-
Domain_Routing#CIDR_notation
    IpRanges: [{ CidrIp: `${ipAddress}/32` }],
  },
],
});

try {
  const { SecurityGroupRules } = await client.send(command);
  console.log(JSON.stringify(SecurityGroupRules, null, 2));
} catch (caught) {
  if (caught instanceof Error && caught.name === "InvalidGroupId.Malformed") {
    console.warn(`${caught.message}. Please provide a valid GroupId.`);
  } else {
    throw caught;
  }
}
};
```

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の「[AuthorizeSecurityGroupIngress](#)」を参照してください。

Kotlin

SDK for Kotlin

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[AuthorizeSecurityGroupIngress](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、EC2-VPC のセキュリティグループの進入ルールを定義します。これらのルールは、SSH (ポート 22) と RDC (ポート 3389) の特定の IP アドレスへのアクセスを許可します。セキュリティグループの名前ではなく、セキュリティグループの ID を使用して、EC2-VPC のセキュリティグループを識別する必要があります。この例で使用される構文には、PowerShell バージョン 3 以降が必要です。

```
$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 )
```

例 2: PowerShell バージョン 2 では、New-Object を使用して IpPermission オブジェクトを作成する必要があります。

```
$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 )
```

例 3: この例では、EC2-Classic のセキュリティグループの進入ルールを定義します。これらのルールは、SSH (ポート 22) と RDC (ポート 3389) の特定の IP アドレスへのアクセスを許可します。この例で使用される構文には、PowerShell バージョン 3 以降が必要です。

```
$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 )
```

例 4: PowerShell バージョン 2 では、New-Object を使用して IpPermission オブジェクトを作成する必要があります。

```
$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 )
```

例 5: この例では、指定されたソースセキュリティグループ (sg-1a2b3c4d) から指定されたセキュリティグループ (sg-12345678) への TCP ポート 8081 のアクセスを許可します。

```
$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 } )
```

例 6: この例では、セキュリティグループ sg-1234abcd の進入ルールに、TCP ポート 22 トラフィック用の CIDR 5.5.5.5/32 を説明付きで追加します。

```
$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
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[AuthorizeSecurityGroupIngress](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、EC2-VPC のセキュリティグループの進入ルールを定義します。これらのルールは、SSH (ポート 22) と RDC (ポート 3389) の特定の IP アドレスへのアクセスを許可します。セキュリティグループの名前ではなく、セキュリティグループの ID を使用して、EC2-VPC のセキュリティグループを識別する必要があります。この例で使用される構文には、PowerShell バージョン 3 以降が必要です。

```
$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 )
```

例 2: PowerShell バージョン 2 では、New-Object を使用して IpPermission オブジェクトを作成する必要があります。

```
$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 )
```

例 3: この例では、EC2-Classic のセキュリティグループの進入ルールを定義します。これらのルールは、SSH (ポート 22) と RDC (ポート 3389) の特定の IP アドレスへのアクセスを許可します。この例で使用される構文には、PowerShell バージョン 3 以降が必要です。

```
$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 )
```

例 4: PowerShell バージョン 2 では、New-Object を使用して IpPermission オブジェクトを作成する必要があります。

```
$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 )
```

例 5: この例では、指定されたソースセキュリティグループ (sg-1a2b3c4d) から指定されたセキュリティグループ (sg-12345678) への TCP ポート 8081 のアクセスを許可します。

```
$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 } )
```

例 6: この例では、セキュリティグループ sg-1234abcd の進入ルールに、TCP ポート 22 トラフィック用の CIDR 5.5.5.5/32 を説明付きで追加します。

```
$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
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[AuthorizeSecurityGroupIngress](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
        security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a
high-level identifier
```

```

        that represents the security group.

        """
        self.ec2_client = ec2_client
        self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.

        :return: An instance of SecurityGroupWrapper initialized with the default
        EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def authorize_ingress(self, ssh_ingress_ip: str) -> Optional[Dict[str, Any]]:
        """
        Adds a rule to the security group to allow access to SSH.

        :param ssh_ingress_ip: The IP address that is granted inbound access to
        connect
                                to port 22 over TCP, used for SSH.
        :return: The response to the authorization request. The 'Return' field of
        the
                response indicates whether the request succeeded or failed, or
        None if no security group is set.
        :raise Handles AWS SDK service-level ClientError, with special handling
        for ResourceAlreadyExists
        """
        if self.security_group is None:
            logger.info("No security group to update.")
            return None

        try:
            ip_permissions = [
                {
                    # SSH ingress open to only the specified IP address.
                    "IpProtocol": "tcp",
                    "FromPort": 22,
                    "ToPort": 22,
                    "IpRanges": [{"CidrIp": f"{ssh_ingress_ip}/32"}],
                }
            ]

```

```

    ]
    response = self.ec2_client.authorize_security_group_ingress(
        GroupId=self.security_group, IpPermissions=ip_permissions
    )
except ClientError as err:
    if err.response["Error"]["Code"] == "InvalidPermission.Duplicate":
        logger.error(
            f"The SSH ingress rule for IP {ssh_ingress_ip} already
exists"

            f"in security group '{self.security_group}'."
        )
        raise
    else:
        return response

```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[AuthorizeSecurityGroupIngress](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

/// Add an ingress rule to a security group explicitly allowing IPv4 address
/// as {ip}/32 over TCP port 22.
pub async fn authorize_security_group_ssh_ingress(
    &self,
    group_id: &str,
    ingress_ips: Vec<Ipv4Addr>,
) -> Result<(), EC2Error> {
    tracing::info!("Authorizing ingress for security group {group_id}");
    self.client
        .authorize_security_group_ingress()
        .group_id(group_id)

```



```

        .set_ip_permissions(Some(
            ingress_ips
                .into_iter()
                .map(|ip| {
                    IpPermission::builder()
                        .ip_protocol("tcp")
                        .from_port(22)
                        .to_port(22)
                        .ip_ranges(IpRange::builder().cidr_ip(format!(
                            "{ip}/32"))).build())
                        .build()
                })
            .collect(),
        ))
        .send()
        .await?;
    Ok(())
}

```

- APIの詳細については、「AWS SDK for Rust API リファレンス」の「[AuthorizeSecurityGroupIngress](#)」を参照してください。

SAP ABAP

SDK for SAP ABAP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

" 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.

```

- APIの詳細については、AWS SDK for SAP ABAP API リファレンスの[AuthorizeSecurityGroupIngress](#)を参照してください。

Swift

SDK for Swift

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

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
    }
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[AuthorizeSecurityGroupIngress](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で `CancelCapacityReservation` を使用する

次のサンプルコードは、`CancelCapacityReservation` を使用する方法を説明しています。

CLI

AWS CLI

キャパシティ予約をキャンセルするには

次の `cancel-capacity-reservation` の例では、指定したキャパシティ予約をキャンセルします。

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

出力:

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

詳細については、Amazon EC2 ユーザーガイドの「[キャパシティ予約のキャンセル](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CancelCapacityReservation](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、キャパシティ予約 `cr-0c1f2345db6f7cdba` をキャンセルします。

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

出力:

```
Confirm  
Are you sure you want to perform this action?
```

```
Performing the operation "Remove-EC2CapacityReservation
(CancelCapacityReservation)" on target "cr-0c1f2345db6f7cdba".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): y
True
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CancelCapacityReservation](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、キャパシティ予約 cr-0c1f2345db6f7cdba をキャンセルします。

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

出力:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2CapacityReservation
(CancelCapacityReservation)" on target "cr-0c1f2345db6f7cdba".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): y
True
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の[CancelCapacityReservation](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **CancelImportTask** を使用する

次のサンプルコードは、CancelImportTask を使用する方法を説明しています。

CLI

AWS CLI

インポートタスクをキャンセルするには

次の `cancel-import-task` の例では、指定したイメージのインポートタスクをキャンセルします。

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

出力:

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

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CancelImportTask](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたインポートタスク (スナップショットまたはイメージのインポート) をキャンセルします。必要に応じて、**-CancelReason** パラメータを使用して理由を指定できます。

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CancelImportTask](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたインポートタスク (スナップショットまたはイメージのインポート) をキャンセルします。必要に応じて、**-CancelReason** パラメータを使用して理由を指定できます。

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CancelImportTask](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で `CancelSpotFleetRequests` を使用する

次のサンプルコードは、`CancelSpotFleetRequests` を使用する方法を説明しています。

CLI

AWS CLI

例 1: スポットフリートリクエストをキャンセルし、関連付けられたインスタスを終了するには

次の `cancel-spot-fleet-requests` の例では、スポットフリートリクエストをキャンセルし、関連付けられたオンデマンドインスタンスとスポットインスタスを終了します。

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

出力:

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

例 2: 関連付けられたインスタスを終了せずに、スポットフリートリクエストをキャンセルするには

次の `cancel-spot-fleet-requests` の例では、関連付けられたオンデマンドインスタンスとスポットインスタスを終了しないで、スポットフリートリクエストをキャンセルします。

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

出力:

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

詳細については、Amazon EC2 ユーザーガイドの「[スポットフリートリクエストのキャンセル](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CancelSpotFleetRequests](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたスポットフリートリクエストをキャンセルし、関連付けられたスポットインスタンスを終了します。

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

例 2: この例では、関連付けられたスポットインスタンスを終了せずに、指定されたスポットフリートリクエストをキャンセルします。

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CancelSpotFleetRequests](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたスポットフリートリクエストをキャンセルし、関連付けられたスポットインスタンスを終了します。

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

例 2: この例では、関連付けられたスポットインスタンスを終了せずに、指定されたスポットフリートリクエストをキャンセルします。

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の [CancelSpotFleetRequests](#) を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください [AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **CancelSpotInstanceRequests** を使用する

次のサンプルコードは、CancelSpotInstanceRequests を使用方法を説明しています。

CLI

AWS CLI

スポットインスタンスリクエストをキャンセルするには

このコマンド例では、スポットインスタンスリクエストをキャンセルします。

コマンド:

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

出力:

```
{
  "CancelledSpotInstanceRequests": [
```

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

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[CancelSpotInstanceRequests](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたスポットインスタンスリクエストをキャンセルします。

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

出力:

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

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CancelSpotInstanceRequests](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたスポットインスタンスリクエストをキャンセルします。

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

出力:

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

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CancelSpotInstanceRequests](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **ConfirmProductInstance** を使用する

次のサンプルコードは、ConfirmProductInstance を使用する方法を説明しています。

CLI

AWS CLI

製品インスタンスを確認するには

この例では、指定された製品コードが指定されたインスタンスに関連付けられているかどうかを判断します。

コマンド:

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

出力:

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

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ConfirmProductInstance](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定された製品コードが指定されたインスタンスに関連付けられているかどうかを判断します。

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ConfirmProductInstance](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定された製品コードが指定されたインスタンスに関連付けられているかどうかを判断します。

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[ConfirmProductInstance](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **CopyImage** を使用する

次のサンプルコードは、CopyImage を使用する方法を説明しています。

CLI

AWS CLI

例 1: AMI を別のリージョンにコピーするには

次の copy-image の例では、指定した AMI を us-west-2 リージョンから us-east-1 リージョンにコピーし、簡単な説明を追加します。

```
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."
```

出力:

```
{
```

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

詳細については、Amazon EC2 ユーザーガイドの「[AMI のコピー](#)」を参照してください。

例 2: AMI を別のリージョンにコピーし、バックアップスナップショットを暗号化するには

次の `copy-image` コマンドは、指定した AMI を `us-west-2` リージョンから現在のリージョンにコピーし、指定した KMS キーを使用してバックアップするスナップショットを暗号化します。

```
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
```

出力:

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

詳細については、Amazon EC2 ユーザーガイドの「[AMI のコピー](#)」を参照してください。

例 3: AMI のコピー時にユーザー定義の AMI タグを含めるには

次の `copy-image` コマンドは、`--copy-image-tags` パラメータを使用して、AMI のコピー時にユーザー定義の AMI タグをコピーします。

```
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
```

出力:

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

詳細については、Amazon EC2 ユーザーガイドの「[AMI のコピー](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CopyImage](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、「欧州 (アイルランド)」リージョンの指定 AMI を「米国西部 (オレゴン)」リージョンにコピーします。-Region が指定されていない場合、現在のデフォルトリージョンが送信先リージョンとして使用されます。

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

出力:

```
ami-87654321
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CopyImage](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、「欧州 (アイルランド)」リージョンの指定 AMI を「米国西部 (オレゴン)」リージョンにコピーします。-Region が指定されていない場合、現在のデフォルトリージョンが送信先リージョンとして使用されます。

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

出力:

```
ami-87654321
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CopyImage](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で CopySnapshot を使用する

次のサンプルコードは、CopySnapshot を使用する方法を説明しています。

CLI

AWS CLI

例 1: スナップショットを別のリージョンにコピーするには

次の copy-snapshot の例では、指定したスナップショットを us-west-2 リージョンから us-east-1 リージョンにコピーし、簡単な説明を追加します。

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

出力:

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

例 2: 暗号化されていないスナップショットをコピーし、新しいスナップショットを暗号化するには

次の copy-snapshot コマンドは、指定した暗号化されていないスナップショットを us-west-2 リージョンから現在のリージョンにコピーし、指定した KMS キーを使用してバックアップするスナップショットを暗号化します。

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

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

出力:

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

詳細については、Amazon EBS ユーザーガイドの「[Amazon EBS スナップショットのコピー](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CopySnapshot](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたスナップショットを欧州 (アイルランド) リージョンから米国西部 (オレゴン) リージョンにコピーします。

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

例 2: デフォルトのリージョンを設定し、Region パラメータを省略すると、デフォルトの送信先リージョンがデフォルトのリージョンになります。

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CopySnapshot](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたスナップショットを欧州 (アイルランド) リージョンから米国西部 (オレゴン) リージョンにコピーします。


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

例 2: デフォルトのリージョンを設定し、Region パラメータを省略すると、デフォルトの送信先リージョンがデフォルトのリージョンになります。

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CopySnapshot](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **CreateCapacityReservation** を使用する

次のサンプルコードは、CreateCapacityReservation を使用する方法を説明しています。

CLI

AWS CLI

例 1: キャパシティ予約を作成するには

次の create-capacity-reservation の例では、eu-west-1a アベイラビリティゾーンにキャパシティ予約を作成します。そこでは、Linux/Unix オペレーティングシステムを実行している 3 つの t2.medium インスタンスを起動できます。デフォルトでは、キャパシティ予約はオープンインスタンス一致基準で作成され、エフェメラルストレージはサポートされません。手動でキャンセルされるまでアクティブな状態を維持します。

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

出力:

```
{
  "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"
  }
}
```

例 2: 指定した日付/時刻に自動的に終了するキャパシティ予約を作成するには

次の create-capacity-reservation の例では、eu-west-1a アベイラビリティーゾーンにキャパシティ予約を作成します。そこでは、Linux/Unix オペレーティングシステムを実行している 3 つの m5.large インスタンスを起動できます。このキャパシティ予約は、2019/8/31 の 23:59:59 に自動的に終了します。

```
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
```

出力:

```
{
  "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"
  }
}

```

例 3: 対象インスタンスの起動のみを受け入れるキャパシティ予約を作成するには

次の `create-capacity-reservation` の例では、対象インスタンスの起動のみを承諾するキャパシティ予約を作成します。

```

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

```

出力:

```

{
  "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"
  }
}

```

```
}
```

詳細については、Amazon EC2 ユーザーガイドの「[キャパシティ予約の作成](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateCapacityReservation](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定された属性で新しいキャパシティ予約を作成します。

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

出力:

```
AvailabilityZone      : eu-west-1b  
AvailableInstanceCount : 2  
CapacityReservationId : cr-0c1f2345db6f7cdba  
CreateDate            : 3/28/2019 9:29:41 AM  
EbsOptimized          : True  
EndDate               : 1/1/0001 12:00:00 AM  
EndDateType           : unlimited  
EphemeralStorage      : False  
InstanceMatchCriteria : open  
InstancePlatform      : Windows  
InstanceType          : m4.xlarge  
State                 : active  
Tags                  : {}  
Tenancy               : default  
TotalInstanceCount    : 2
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CreateCapacityReservation](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定された属性で新しいキャパシティ予約を作成します。

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

出力:

```
AvailabilityZone      : eu-west-1b
AvailableInstanceCount : 2
CapacityReservationId : cr-0c1f2345db6f7cdba
CreateDate            : 3/28/2019 9:29:41 AM
EbsOptimized          : True
EndDate               : 1/1/0001 12:00:00 AM
EndDateType           : unlimited
EphemeralStorage      : False
InstanceMatchCriteria : open
InstancePlatform      : Windows
InstanceType          : m4.xlarge
State                 : active
Tags                  : {}
Tenancy                : default
TotalInstanceCount    : 2
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CreateCapacityReservation](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **CreateCustomerGateway** を使用する

次のサンプルコードは、CreateCustomerGateway を使用する方法を説明しています。

CLI

AWS CLI

カスタマーゲートウェイを作成するには

この例では、外部インターフェイスに指定された IP アドレスを持つカスタマーゲートウェイを作成します。

コマンド:

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

出力:

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

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateCustomerGateway](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたカスタマーゲートウェイを作成します。

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

出力:

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CreateCustomerGateway](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたカスタマーゲートウェイを作成します。

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

出力:

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CreateCustomerGateway](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **CreateDhcpOptions** を使用する

次のサンプルコードは、CreateDhcpOptions を使用する方法を説明しています。

CLI

AWS CLI

DHCP オプションセットを作成するには

次の create-dhcp-options の例では、ドメイン名、ドメインネームサーバー、NetBIOS ノードタイプを指定する DHCP オプションセットを作成します。

```
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"
```

出力:

```
{
```

```
"DhcpOptions": {
  "DhcpConfigurations": [
    {
      "Key": "domain-name",
      "Values": [
        {
          "Value": "example.com"
        }
      ]
    },
    {
      "Key": "domain-name-servers",
      "Values": [
        {
          "Value": "10.2.5.1"
        },
        {
          "Value": "10.2.5.2"
        }
      ]
    },
    {
      "Key": "netbios-node-type",
      "Values": [
        {
          "Value": "2"
        }
      ]
    }
  ],
  "DhcpOptionsId": "dopt-06d52773eff4c55f3"
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateDhcpOptions](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定された DHCP オプションのセットを作成します。この例で使用する構文には、PowerShell バージョン 3 以降が必要です。


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

出力:

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

例 2: PowerShell バージョン 2 では、New-Object を使用して各 DHCP オプションを作成する必要があります。

```
$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)
```

出力:

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CreateDhcpOptions](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定された DHCP オプションのセットを作成します。この例で使用される構文には、PowerShell バージョン 3 以降が必要です。

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

出力:

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

例 2: PowerShell バージョン 2 では、New-Object を使用して各 DHCP オプションを作成する必要があります。

```
$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)
```

出力:

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CreateDhcpOptions](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **CreateFlowLogs** を使用する

次のサンプルコードは、CreateFlowLogs を使用する方法を説明しています。

CLI

AWS CLI

例 1: フローログを作成するには

次の create-flow-logs の例では、指定したネットワークインターフェイスについて拒否されたすべてのトラフィックをキャプチャするフローログを作成します。フローログは、指定された IAM ロールのアクセス許可を使用して CloudWatch Logs のロググループに配信されます。

```
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
```

出力:

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

詳細については、「Amazon VPC ユーザーガイド」の「[VPC フローログを使用した IP トラフィックのログ記録](#)」を参照してください。

例 2: カスタム形式でフローログを作成するには

次の create-flow-logs の例では、指定した VPC のすべてのトラフィックをキャプチャするフローログを作成し、そのフローログを Amazon S3 バケットに配信します。--log-format パラメータにより、フローログレコードのカスタム形式が指定されます。Windows でこのコマンドを実行するには、一重引用符 (') を二重引用符 (") に変更します。

```
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}'
```

詳細については、「Amazon VPC ユーザーガイド」の「[VPC フローログを使用した IP トラフィックのログ記録](#)」を参照してください。

例 3: 1 分間の最大集計間隔でフローログを作成するには

次の create-flow-logs の例では、指定した VPC のすべてのトラフィックをキャプチャするフローログを作成し、そのフローログを Amazon S3 バケットに配信します。--max-aggregation-interval パラメータは、最大集約間隔を 60 秒 (1 分) に指定します。

```
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
```

詳細については、「Amazon VPC ユーザーガイド」の「[VPC フローログを使用した IP トラフィックのログ記録](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateFlowLogs](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、「Admin」ロールの権限を使用して、サブネット subnet-1d234567 に対する EC2 フローログを、すべての「REJECT」トラフィックを対象に、「subnet1-log」という名前の cloud-watch-log に作成します。

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

出力:

ClientToken	FlowLogIds	Unsuccessful
m1VN2cxP3iB4qo//VUK15EU6cF7gQL0xcqNefvjeTGw=	{f1-012fc34eed5678c9d}	{}

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CreateFlowLogs](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、「Admin」ロールの権限を使用して、サブネット subnet-1d234567 に対する EC2 フローログを、すべての「REJECT」トラフィックを対象に、「subnet1-log」という名前の cloud-watch-log に作成します。

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

出力:

ClientToken	FlowLogIds	Unsuccessful
-----	-----	-----
m1VN2cxP3iB4qo//VUK15EU6cF7gQL0xcqNefvjeTGw=	{f1-012fc34eed5678c9d}	{}

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CreateFlowLogs](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **CreateImage** を使用する

次のサンプルコードは、CreateImage を使用する方法を説明しています。

CLI

AWS CLI

例 1: Amazon EBS-backed インスタンスから AMI を作成するには

次の create-image の例では、指定したインスタンスから AMI を作成します。

```
aws ec2 create-image \  
  --instance-id i-1234567890abcdef0 \  
  --name my-ami
```

```
--name "My server" \  
--description "An AMI for my server"
```

出力:

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

AMI のブロックデバイスマッピングの指定の詳細については、「Amazon EC2 ユーザーガイド」の「[Specifying a block device mapping for an AMI](#)」を参照してください。

例 2: 再起動せずに Amazon EBS-backed インスタンスから AMI を作成するには

次の create-image の例では、AMI を作成し、--no-reboot パラメータを設定して、イメージの作成前にインスタンスが再起動されないようにします。

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

出力:

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

AMI のブロックデバイスマッピングの指定の詳細については、「Amazon EC2 ユーザーガイド」の「[Specifying a block device mapping for an AMI](#)」を参照してください。

例 3: 作成時に AMI とスナップショットにタグを付けるには

次の create-image の例では、AMI を作成し、AMI とスナップショットに同じタグ cost-center=cc123 を付けます。

```
aws ec2 create-image \  
  --instance-id i-1234567890abcdef0 \  
  --name "My server" \  
  --tag-specifications 'TagSpecification={ResourceType=Image,Tags=[{Key=cost-center,Value=cc123}]}
```

```
--tag-specifications "ResourceType=image,Tags=[{Key=cost-center,Value=cc123}]" "ResourceType=snapshot,Tags=[{Key=cost-center,Value=cc123}]"
```

出力:

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

作成時のリソースのタグ付けの詳細については、「Amazon EC2 ユーザーガイド」の「[Add tags on resource creation](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateImage](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたインスタンスから、指定された名前と説明を持つ AMI を作成します。Amazon EC2 は、イメージを作成する前にインスタンスのクリーンシャットダウンを試み、完了時にインスタンスを再起動します。

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

例 2: この例では、指定されたインスタンスから、指定された名前と説明を持つ AMI を作成します。Amazon EC2 は、インスタンスのシャットダウンと再起動をせずにイメージを作成するため、作成されたイメージのファイルシステムの整合性は保証されません。

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

例 3: この例では、3 つのボリュームを持つ AMI を作成します。最初のボリュームは Amazon EBS スナップショットに基づいています。2 番目のボリュームは空の 100 GiB Amazon EBS ボリュームです。3 番目のボリュームはインスタンスストアボリュームです。この例で 사용되는構文には、PowerShell バージョン 3 以降が必要です。

```
$ebsBlock1 = @{$SnapshotId="snap-1a2b3c4d"}
```

```
$ebsBlock2 = @{VolumeSize=100}

New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description
  "My web server AMI" -BlockDeviceMapping @( @{DeviceName="/dev/sdf";Ebs=
  $ebsBlock1}, @{DeviceName="/dev/sdg";Ebs=$ebsBlock2}, @{DeviceName="/dev/
  sdc";VirtualName="ephemeral0"})
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CreateImage](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたインスタンスから、指定された名前と説明を持つ AMI を作成します。Amazon EC2 は、イメージを作成する前にインスタンスのクリーンシャットダウンを試み、完了時にインスタンスを再起動します。

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

例 2: この例では、指定されたインスタンスから、指定された名前と説明を持つ AMI を作成します。Amazon EC2 は、インスタンスのシャットダウンと再起動をせずにイメージを作成するため、作成されたイメージのファイルシステムの整合性は保証されません。

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

例 3: この例では、3 つのボリュームを持つ AMI を作成します。最初のボリュームは Amazon EBS スナップショットに基づいています。2 番目のボリュームは空の 100 GiB Amazon EBS ボリュームです。3 番目のボリュームはインスタンスストアボリュームです。この例で 사용되는構文には、PowerShell バージョン 3 以降が必要です。

```
$ebsBlock1 = @{SnapshotId="snap-1a2b3c4d"}
$ebsBlock2 = @{VolumeSize=100}

New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description
  "My web server AMI" -BlockDeviceMapping @( @{DeviceName="/dev/sdf";Ebs=
  $ebsBlock1}, @{DeviceName="/dev/sdg";Ebs=$ebsBlock2}, @{DeviceName="/dev/
  sdc";VirtualName="ephemeral0"})
```

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CreateImage](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で `CreateInstanceExportTask` を使用する

次のサンプルコードは、`CreateInstanceExportTask` を使用方法を説明しています。

CLI

AWS CLI

インスタンスをエクスポートするには

このコマンド例では、インスタンス `i-1234567890abcdef0` を Amazon S3 バケット `myexportbucket` にエクスポートするタスクを作成します。

コマンド:

```
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
```

出力:

```
{  
  "ExportTask": {  
    "State": "active",  
    "InstanceExportDetails": {  
      "InstanceId": "i-1234567890abcdef0",  
      "TargetEnvironment": "vmware"  
    },  
    "ExportToS3Task": {  
      "S3Bucket": "myexportbucket",  
      "S3Key": "RHEL5export-i-fh8sjjsq.ova",  
      "DiskImageFormat": "vmdk",  
      "ContainerFormat": "ova"  
    },  
    "Description": "RHEL5 instance",  
    "ExportTaskId": "export-i-fh8sjjsq"  
  }  
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateInstanceExportTask](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、停止したインスタンス **i-0800b00a00EXAMPLE** を仮想ハードディスク (VHD) として S3 バケット **testbucket-export-instances-2019** にエクスポートします。ターゲット環境は **Microsoft**、インスタンスが **us-east-1** リージョンにあり、ユーザーのデフォルト AWS リージョンが **us-east-1** ではないため、リージョンパラメータが追加されます。エクスポートタスクのステータスを取得するには、このコマンドの結果から **ExportTaskId** 値をコピーし、**Get-EC2ExportTask -ExportTaskId export_task_ID_from_results.** を実行します。

```
New-EC2InstanceExportTask -InstanceId i-0800b00a00EXAMPLE -  
ExportToS3Task_DiskImageFormat VHD -ExportToS3Task_S3Bucket "amzn-s3-demo-bucket"  
-TargetEnvironment Microsoft -Region us-east-1
```

出力:

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CreateInstanceExportTask](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、停止したインスタンス **i-0800b00a00EXAMPLE** を仮想ハードディスク (VHD) として S3 バケット **testbucket-export-instances-2019** にエクスポートします。ターゲット環境は **Microsoft**、インスタンスが **us-east-1** リージョンにあり、ユーザーのデフォルト AWS リージョンが **us-east-1** ではないため、リージョンパラメータが追加されます。エクスポートタスクのステータスを取得するには、このコマンドの結果から **ExportTaskId** 値をコピーし、**Get-EC2ExportTask -ExportTaskId export_task_ID_from_results.** を実行します。

```
New-EC2InstanceExportTask -InstanceId i-0800b00a00EXAMPLE -  
ExportToS3Task_DiskImageFormat VHD -ExportToS3Task_S3Bucket "amzn-s3-demo-bucket"  
-TargetEnvironment Microsoft -Region us-east-1
```

出力:

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CreateInstanceExportTask](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **CreateInternetGateway** を使用する

次のサンプルコードは、CreateInternetGateway を使用方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [プライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成する](#)
- [Amazon VPC の使用を開始する](#)

CLI

AWS CLI

インターネットゲートウェイを作成するには

次の create-internet-gateway の例では、タグ Name=my-igw を持つインターネットゲートウェイを作成します。

```
aws ec2 create-internet-gateway \
```

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

出力:

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

詳細については、Amazon VPC ユーザーガイドの「[インターネットゲートウェイ](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateInternetGateway](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、インターネットゲートウェイを作成します。

```
New-EC2InternetGateway
```

出力:

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CreateInternetGateway](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、インターネットゲートウェイを作成します。

```
New-EC2InternetGateway
```

出力:

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CreateInternetGateway](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **CreateKeyPair**で を使用する

次のサンプルコードは、CreateKeyPair を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)
- [Amazon VPC の使用を開始する](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

        var response = await _amazonEC2.CreateKeyPairAsync(request);

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

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

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

        throw;
    }
}
```

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

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

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

    return pemFileName;
}
```

- API の詳細については、AWS SDK for .NET API リファレンスの「[CreateKeyPair](#)」を参照してください。

Bash

AWS CLI Bash スクリプトを使用する

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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



```

        ;;
    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
}

```

この例で使用されているユーティリティ関数。

```

#####
# 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
}

```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[CreateKeyPair](#)」を参照してください。

C++

SDK for C++

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

//! Create an Amazon Elastic Compute Cloud (Amazon EC2) instance key pair.
/*!
    \param keyPairName: A name for a key pair.
    \param keyFilePath: File path where the credentials are stored. Ignored if it
    is an empty string;
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::EC2::createKeyPair(const Aws::String &keyPairName, const Aws::String
&keyFilePath,
                                const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::CreateKeyPairRequest request;
    request.SetKeyName(keyPairName);

    Aws::EC2::Model::CreateKeyPairOutcome outcome =
ec2Client.CreateKeyPair(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to create key pair - " << keyPairName << ". " <<
        outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully created key pair named " <<
        keyPairName << std::endl;
        if (!keyFilePath.empty()) {
            std::ofstream keyFile(keyFilePath.c_str());
            keyFile << outcome.GetResult().GetKeyMaterial();
            keyFile.close();
            std::cout << "Keys written to the file " <<
            keyFilePath << std::endl;
        }
    }
}

```

```
    }  
  
    return outcome.IsSuccess();  
}
```

- API の詳細については、AWS SDK for C++ API リファレンスの「[CreateKeyPair](#)」を参照してください。

CLI

AWS CLI

キーペアを作成するには

この例では、MyKeyPair という名前のキーペアが作成されます。

コマンド:

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

出力は ASCII バージョンのプライベートキーとキーフィンガープリントです。キーはファイルに保存する必要があります。

詳細については、「AWS コマンドラインインターフェイスユーザーガイド」でキーペアの使用方法を参照してください。

- API の詳細については、「AWS CLI Command Reference」の「[CreateKeyPair](#)」を参照してください。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

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

    return responseFuture;
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[CreateKeyPair](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

/**
 * Creates an ED25519 or 2048-bit RSA key pair with the specified name and in the
 * specified PEM or PPK format.
 * Amazon EC2 stores the public key and displays the private key for you to save
 * to a file.
 * @param {{ keyName: string }} options
 */
export const main = async ({ keyName }) => {
  const client = new EC2Client({});
  const command = new CreateKeyPairCommand({
    KeyName: keyName,
  });

  try {
    const { KeyMaterial, KeyName } = await client.send(command);
    console.log(KeyName);
    console.log(KeyMaterial);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidKeyPair.Duplicate") {
      console.warn(`${caught.message}. Try another key name.`);
    } else {
      throw caught;
    }
  }
};
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[CreateKeyPair](#)」を参照してください。

Kotlin

SDK for Kotlin

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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}")  
    }  
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[CreateKeyPair](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、キーペアを作成し、指定した名前のファイルに PEM エンコードされた RSA プライベートキーをキャプチャします。PowerShell を使用している場合は、有効なキーを生成するためにエンコーディングを `ascii` に設定する必要があります。詳細については、「[コマンドラインインターフェイスユーザーガイド](#)」の Amazon EC2 AWS キーペアの作成、表示、削除 (<https://docs.aws.amazon.com/cli/latest/userguide/cli-services-ec2-keypairs.html>)」を参照してください。

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CreateKeyPair](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、キーペアを作成し、指定した名前のファイルに PEM エンコードされた RSA プライベートキーをキャプチャします。PowerShell を使用している場合は、有効なキーを生成するためにエンコーディングを `ascii` に設定する必要があります。詳細については、「[コマンドラインインターフェイスユーザーガイド](#)」の Amazon EC2 AWS キーペアの作成、表示、削除 (<https://docs.aws.amazon.com/cli/latest/userguide/cli-services-ec2-keypairs.html>)」を参照してください。

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CreateKeyPair](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
class KeyPairWrapper:
    """
    Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions.
    This class provides methods to create, list, and delete EC2 key pairs.
    """

    def __init__(
        self,
        ec2_client: boto3.client,
        key_file_dir: Union[tempfile.TemporaryDirectory, str],
        key_pair: Optional[dict] = None,
    ):
        """
```



```

        Initializes the KeyPairWrapper with the specified EC2 client, key file
        directory,
        and an optional key pair.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param key_file_dir: The folder where the private key information is
stored.
                           This should be a secure folder.
        :param key_pair: A dictionary representing the Boto3 KeyPair object.
                           This is a high-level object that wraps key pair actions.
Optional.
    """
    self.ec2_client = ec2_client
    self.key_pair = key_pair
    self.key_file_path: Optional[str] = None
    self.key_file_dir = key_file_dir

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

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

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

        :param key_name: The name of the key pair to create.
        :return: A dictionary representing the Boto3 KeyPair object that
represents the newly created key pair.

```

```

        :raises ClientError: If there is an error in creating the key pair, for
        example, if a key pair with the same name already exists.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_name)
            self.key_pair = response
            self.key_file_path = os.path.join(
                self.key_file_dir.name, f"{self.key_pair['KeyName']}.pem"
            )
            with open(self.key_file_path, "w") as key_file:
                key_file.write(self.key_pair["KeyMaterial"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidKeyPair.Duplicate":
                logger.error(
                    f"A key pair called {key_name} already exists. "
                    "Please choose a different name for your key pair "
                    "or delete the existing key pair before creating."
                )
            raise
        else:
            return self.key_pair

```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[CreateKeyPair](#)」を参照してください。

Ruby

SDK for Ruby

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

# 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
  rescue StandardError => e
    puts "Error getting information about key pairs: #{e.message}"
  end

# Deletes a key pair in Amazon Elastic Compute Cloud (Amazon EC2).
#
# Prerequisites:
#
# - The key pair to delete.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param key_pair_name [String] The name of the key pair to delete.
# @return [Boolean] true if the key pair was deleted; otherwise, false.
# @example
#   exit 1 unless key_pair_deleted?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-key-pair'
#   )
def key_pair_deleted?(ec2_client, key_pair_name)
  ec2_client.delete_key_pair(key_name: key_pair_name)
  true
rescue StandardError => e
  puts "Error deleting key pair: #{e.message}"
  false
end

# Example usage:
def run_me
  key_pair_name = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:  ruby ec2-ruby-example-key-pairs.rb KEY_PAIR_NAME REGION'
    puts 'Example: ruby ec2-ruby-example-key-pairs.rb my-key-pair us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    key_pair_name = 'my-key-pair'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.

```

```
else
  key_pair_name = ARGV[0]
  region = ARGV[1]
end

ec2_client = Aws::EC2::Client.new(region: region)

puts 'Displaying existing key pair names before creating this key pair...'
describe_key_pairs(ec2_client)

puts '-' * 10
puts 'Creating key pair...'
unless key_pair_created?(ec2_client, key_pair_name)
  puts 'Stopping program.'
  exit 1
end

puts '-' * 10
puts 'Displaying existing key pair names after creating this key pair...'
describe_key_pairs(ec2_client)

puts '-' * 10
puts 'Deleting key pair...'
unless key_pair_deleted?(ec2_client, key_pair_name)
  puts 'Stopping program. You must delete the key pair yourself.'
  exit 1
end
puts 'Key pair deleted.'

puts '-' * 10
puts 'Now that the key pair is deleted, ' \
  'also deleting the related private key pair file...'
filename = File.join(Dir.home, "#{key_pair_name}.pem")
File.delete(filename)
if File.exist?(filename)
  puts "Could not delete file at '#{filename}'. You must delete it yourself."
else
  puts 'File deleted.'
end

puts '-' * 10
puts 'Displaying existing key pair names after deleting this key pair...'
describe_key_pairs(ec2_client)
end
```

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

- API の詳細については、AWS SDK for Ruby API リファレンスの「[CreateKeyPair](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

EC2 クライアントの `create_key_pair` を呼び出し、返されたマテリアルを抽出する Rust 実装。

```
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))
}
```

`create_key_impl` を呼び出し、PEM プライベートキーを安全に保存する関数。

```
/// 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)
    }

```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[CreateKeyPair](#)」を参照してください。

SAP ABAP

SDK for SAP ABAP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
TRY.  
    oo_result = lo_ec2->createkeypair( iv_keyname = iv_key_name ).  
    " oo_result is returned for testing purposes. "  
    MESSAGE 'Amazon EC2 key pair created.' TYPE 'I'.  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception->av_err_msg }|.  
    MESSAGE lv_error TYPE 'E'.  
ENDTRY.
```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「[CreateKeyPair](#)」を参照してください。

Swift

SDK for Swift

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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
    }
}
```

- APIの詳細については、「AWS SDK for Swift API リファレンス」の「[CreateKeyPair](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI `CreateLaunchTemplate`で を使用する

次のサンプルコードは、`CreateLaunchTemplate` を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [レジリエントなサービスの構築と管理](#)
- [プライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成する](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling.
/// The launch template specifies a Bash script in its user data field that
runs after
/// the instance is started. This script installs the Python packages and
starts a Python
/// web server on the instance.
/// </summary>
/// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
/// <param name="instancePolicyPath">The path to a permissions policy to
create and attach to the profile.</param>
/// <returns>The template object.</returns>
public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
{
    try
    {
        await CreateKeyPair(_keyPairName);
```

```

        await CreateInstanceProfileWithName(_instancePolicyName,
            _instanceRoleName,
            _instanceProfileName, instancePolicyPath);

        var startServerText = await File.ReadAllTextAsync(startupScriptPath);
        var plainTextBytes =
            System.Text.Encoding.UTF8.GetBytes(startServerText);

        var amiLatest = await _amazonSsm.GetParameterAsync(
            new GetParameterRequest() { Name = _amiParam });
        var amiId = amiLatest.Parameter.Value;
        var launchTemplateResponse = await
            _amazonEc2.CreateLaunchTemplateAsync(
                new CreateLaunchTemplateRequest()
                {
                    LaunchTemplateName = _launchTemplateName,
                    LaunchTemplateData = new RequestLaunchTemplateData()
                    {
                        InstanceType = _instanceType,
                        ImageId = amiId,
                        IamInstanceProfile =
                            new
                                LaunchTemplateIamInstanceProfileSpecificationRequest()
                                {
                                    Name = _instanceProfileName
                                },
                        KeyName = _keyPairName,
                        UserData = System.Convert.ToBase64String(plainTextBytes)
                    }
                });
        return launchTemplateResponse.LaunchTemplate;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode ==
            "InvalidLaunchTemplateName.AlreadyExistsException")
        {
            _logger.LogError($"Could not create the template, the name
                {_launchTemplateName} already exists. " +
                    $"Please try again with a unique name.");
        }

        throw;
    }

```

```

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

```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[CreateLaunchTemplate](#)」を参照してください。

CLI

AWS CLI

例 1: 起動テンプレートを作成するには

次の `create-launch-template` の例では、インスタンスを起動し、インスタンスにパブリック IP アドレスと IPv6 アドレスを割り当て、インスタンスのタグを作成するサブネットを指定する起動テンプレートを作成しています。

```

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"}]}]}'

```

出力:

```

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

```

}

詳細については、「Amazon Elastic Compute Cloud ユーザーガイド」の「起動テンプレートからのインスタンスの起動」を参照してください。JSON 形式のパラメータで引用する方法については、「AWS コマンドラインインターフェイスユーザーガイド」の「文字列の引用」を参照してください。

例 2: Amazon EC2 Auto Scaling の起動テンプレートを作成するには

次の create-launch-template の例では、複数のタグとブロックデバイスマッピングを使ってインスタンス起動時に追加の EBS ボリュームを指定する起動テンプレートを作成しています。Auto Scaling グループがインスタンスを起動する VPC のセキュリティグループに対応する Groups の値を指定します。Auto Scaling グループのプロパティとして VPC とサブネットを指定します。

```
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
```

出力:

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

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Auto Scaling グループの起動テンプレートを作成する」を参照してください。JSON 形式のパラメータで引用する方法については、「AWS コマンドラインインターフェイスユーザーガイド」の「文字列の引用」を参照してください。

例 3: EBS ボリュームの暗号化を指定する起動テンプレートを作成するには

次の `create-launch-template` の例では、暗号化されていないスナップショットから作成された暗号化された EBS ボリュームを含む起動テンプレートを作成しています。また、作成時にボリュームにタグ付けしています。暗号化がデフォルトで無効になっている場合、次の例のように `"Encrypted"` オプションを指定する必要があります。`"KmsKeyId"` オプションを使用してカスタマー管理の CMK を指定する場合は、デフォルトで暗号化が有効になっていても `"Encrypted"` オプションを指定する必要があります。

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

`config.json` の内容:

```
{  
  "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"
      }
    ]
  }
}

```

出力:

```

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

```

詳細については、「Amazon Elastic Compute Cloud ユーザーガイド」の「スナップショットからの Amazon EBS ボリュームの復元」と「デフォルトでの暗号化」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateLaunchTemplate](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

const ssmClient = new SSMClient({});
const { Parameter } = await ssmClient.send(
  new GetParameterCommand({
    Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
  }),

```

```
);  
const ec2Client = new EC2Client({});  
await ec2Client.send(  
  new CreateLaunchTemplateCommand({  
    LaunchTemplateName: NAMES.launchTemplateName,  
    LaunchTemplateData: {  
      InstanceType: "t3.micro",  
      ImageId: Parameter.Value,  
      IamInstanceProfile: { Name: NAMES.instanceProfileName },  
      UserData: readFileSync(  
        join(RESOURCES_PATH, "server_startup_script.sh"),  
      ).toString("base64"),  
      KeyName: NAMES.keyPairName,  
    },  
  }),  
),
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[CreateLaunchTemplate](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

この例では、インスタンスに特定のアクセス許可を付与するインスタンスプロファイルと、起動後にインスタンスで実行されるユーザーデータの Bash スクリプトを含む起動テンプレートを作成します。

```
class AutoScalingWrapper:  
    """  
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.  
    """  
  
    def __init__(  
        self,
```



```

        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

```

```

def create_template(
    self, server_startup_script_file: str, instance_policy_file: str
) -> Dict[str, Any]:
    """
    Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
    Scaling. The
    launch template specifies a Bash script in its user data field that runs
    after
    the instance is started. This script installs Python packages and starts
    a
    Python web server on the instance.

    :param server_startup_script_file: The path to a Bash script file that is
    run
                                     when an instance starts.
    :param instance_policy_file: The path to a file that defines a
    permissions policy
                               to create and attach to the instance
    profile.
    :return: Information about the newly created template.
    """
    template = {}
    try:
        # Create key pair and instance profile
        self.create_key_pair(self.key_pair_name)
        self.create_instance_profile(
            instance_policy_file,
            self.instance_policy_name,
            self.instance_role_name,
            self.instance_profile_name,
        )

        # Read the startup script
        with open(server_startup_script_file) as file:
            start_server_script = file.read()

        # Get the latest AMI ID
        ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
        ami_id = ami_latest["Parameter"]["Value"]

        # Create the launch template
        lt_response = self.ec2_client.create_launch_template(
            LaunchTemplateName=self.launch_template_name,

```

```

        LaunchTemplateData={
            "InstanceType": self.inst_type,
            "ImageId": ami_id,
            "IamInstanceProfile": {"Name": self.instance_profile_name},
            "UserData": base64.b64encode(
                start_server_script.encode(encoding="utf-8")
            ).decode(encoding="utf-8"),
            "KeyName": self.key_pair_name,
        },
    )
    template = lt_response["LaunchTemplate"]
    log.info(
        f"Created launch template {self.launch_template_name} for AMI {ami_id} on {self.inst_type}."
    )
    except ClientError as err:
        log.error(f"Failed to create launch template {self.launch_template_name}.")
        error_code = err.response["Error"]["Code"]
        if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
            log.info(
                f"Launch template {self.launch_template_name} already exists, nothing to do."
            )
        log.error(f"Full error:\n\t{err}")
    return template

```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[CreateLaunchTemplate](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください [AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **CreateNetworkAcl** を使用する

次のサンプルコードは、CreateNetworkAcl を使用する方法を説明しています。

CLI

AWS CLI

ネットワーク ACL を作成するには

この例では、指定した VPC に対してネットワーク ACL を作成します。

コマンド:

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

出力:

```
{
  "NetworkAcl": {
    "Associations": [],
    "NetworkAclId": "acl-5fb85d36",
    "VpcId": "vpc-a01106c2",
    "Tags": [],
    "Entries": [
      {
        "CidrBlock": "0.0.0.0/0",
        "RuleNumber": 32767,
        "Protocol": "-1",
        "Egress": true,
        "RuleAction": "deny"
      },
      {
        "CidrBlock": "0.0.0.0/0",
        "RuleNumber": 32767,
        "Protocol": "-1",
        "Egress": false,
        "RuleAction": "deny"
      }
    ],
    "IsDefault": false
  }
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateNetworkAcl](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定した VPC に対してネットワーク ACL を作成します。

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

出力:

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CreateNetworkAcl](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定した VPC に対してネットワーク ACL を作成します。

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

出力:

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CreateNetworkAcl](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で `CreateNetworkAclEntry` を使用する

次のサンプルコードは、`CreateNetworkAclEntry` を使用する方法を説明しています。

CLI

AWS CLI

ネットワーク ACL エントリを作成するには

この例では、指定したネットワーク ACL のエントリを作成します。このルールは、UDP ポート 53 (DNS) 上の任意の IPv4 アドレス (0.0.0.0/0) から任意の関連付けられたサブネットへの受信トラフィックを許可します。コマンドが成功した場合、出力は返りません。

コマンド:

```
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
```

この例では、TCP ポート 80 (HTTP) 上の任意の IPv6 アドレス (::/0) からの受信トラフィックを許可する、指定したネットワーク ACL のルールを作成します。

コマンド:

```
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
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateNetworkAclEntry](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定したネットワーク ACL のエントリを作成します。このルールは、UDP ポート 53 (DNS) 上の任意の場所 (0.0.0.0/0) から任意の関連付けられたサブネットへの受信トラフィックを許可します。

```
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
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CreateNetworkAclEntry](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定したネットワーク ACL のエントリを作成します。このルールは、UDP ポート 53 (DNS) 上の任意の場所 (0.0.0.0/0) から任意の関連付けられたサブネットへの受信トラフィックを許可します。

```
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
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CreateNetworkAclEntry](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **CreateNetworkInterface** を使用する

次のサンプルコードは、CreateNetworkInterface を使用する方法を説明しています。

CLI

AWS CLI

例 1: ネットワークインターフェイスの IPv4 アドレスを指定するには

次の `create-network-interface` の例では、指定したプライマリ IPv4 アドレスを持つ指定したサブネット用にネットワークインターフェイスを作成します。

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

出力:

```
{  
  "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"  
  }  
}
```



```
}
```

例 2: IPv4 アドレスと IPv6 アドレスを持つネットワークインターフェイスを作成するには

次の `create-network-interface` の例では、Amazon EC2 で選択されている IPv4 アドレスと IPv6 アドレスを使用して、指定したサブネット用にネットワークインターフェイスを作成します。

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

出力:

```
{  
  "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"
  }
}

```

例 3: 接続追跡設定オプションを使用してネットワークインターフェイスを作成するには

次の `create-network-interface` の例では、ネットワークインターフェイスを作成し、アイドル接続追跡タイムアウトを設定します。

```

aws ec2 create-network-interface \
  --subnet-id subnet-00a24d0d67acf6333 \
  --groups sg-02e57dbcf0331c1b \
  --connection-tracking-specification TcpEstablishedTimeout=86400,UdpTimeout=60

```

出力:

```

{
  "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"
  }
}

```

例 4: Elastic Fabric Adapter を作成するには

次の `create-network-interface` の例では、EFA を作成します。

```

aws ec2 create-network-interface \
  --interface-type efa \
  --subnet-id subnet-00a24d0d67acf6333 \
  --description "my efa" \
  --groups sg-02e57dbcfef0331c1b

```

出力:

```

{
  "NetworkInterface": {
    "AvailabilityZone": "us-west-2a",
    "Description": "my efa",
    "Groups": [
      {
        "GroupName": "my-efa-sg",
        "GroupId": "sg-02e57dbcfef0331c1b"
      }
    ]
  },

```

```

    "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"
  }
}

```

詳細については、「Amazon EC2 ユーザーガイド」の「[Elastic Network Interface](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateNetworkInterface](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたネットワークインターフェイスを作成します。

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

出力:

```
Association      :
```

```

Attachment      :
AvailabilityZone : us-west-2c
Description     : my network interface
Groups         : {my-security-group}
MacAddress      : 0a:72:bc:1a:cd:7f
NetworkInterfaceId : eni-12345678
OwnerId        : 123456789012
PrivateDnsName  : ip-10-0-0-17.us-west-2.compute.internal
PrivateIpAddress : 10.0.0.17
PrivateIpAddresses : {}
RequesterId     :
RequesterManaged : False
SourceDestCheck : True
Status         : pending
SubnetId       : subnet-1a2b3c4d
TagSet         : {}
VpcId         : vpc-12345678

```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CreateNetworkInterface](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたネットワークインターフェイスを作成します。

```

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

```

出力:

```

Association      :
Attachment       :
AvailabilityZone : us-west-2c
Description     : my network interface
Groups         : {my-security-group}
MacAddress      : 0a:72:bc:1a:cd:7f
NetworkInterfaceId : eni-12345678
OwnerId        : 123456789012
PrivateDnsName  : ip-10-0-0-17.us-west-2.compute.internal
PrivateIpAddress : 10.0.0.17
PrivateIpAddresses : {}
RequesterId     :
RequesterManaged : False

```

```
SourceDestCheck    : True
Status             : pending
SubnetId           : subnet-1a2b3c4d
TagSet             : {}
VpcId              : vpc-12345678
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CreateNetworkInterface](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で `CreatePlacementGroup` を使用する

次のサンプルコードは、`CreatePlacementGroup` を使用する方法を説明しています。

CLI

AWS CLI

プレイスメントグループを作成するには

このコマンド例では、指定された名前のプレイスメントグループを作成します。

コマンド:

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

パーティションプレイスメントグループを作成するには

このコマンド例では、HDFS-Group-A という名前のパーティションプレイスメントグループと 5 つのパーティションを作成します。

コマンド:

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

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreatePlacementGroup](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定された名前のプレースメントグループを作成します。

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CreatePlacementGroup](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定された名前のプレースメントグループを作成します。

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CreatePlacementGroup](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **CreateRoute** を使用する

次のサンプルコードは、CreateRoute を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [プライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成する](#)
- [Amazon VPC の使用を開始する](#)
- [Transit Gateway を開始する](#)

CLI

AWS CLI

ルートを作成するには

この例では、指定したルートテーブルに対してルートを作成します。ルートはすべての IPv4 トラフィック (0.0.0.0/0) と一致し、指定されたインターネットゲートウェイにルーティングされます。コマンドが成功した場合、出力は返りません。

コマンド:

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

このコマンド例では、ルートテーブル rtb-g8ff4ea2 にルートを作成します。ルートは IPv4 CIDR ブロック 10.0.0.0/16 のトラフィックに一致し、VPC ピアリング接続 pcx-111aaa22 にルーティングされます。このルートにより、トラフィックを VPC ピアリング接続のピア VPC にルーティングできます。コマンドが成功した場合、出力は返りません。

コマンド:

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

この例では、すべての IPv6 トラフィック (:::/0) に一致するルートを指定されたルートテーブルに作成し、指定された Egress 専用インターネットゲートウェイにルーティングします。

コマンド:

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

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateRoute](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定したルートテーブルに対してルートを作成します。ルートはすべてのトラフィックと一致し、これを指定されたインターネットゲートウェイに送信します。

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


出力:

```
True
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CreateRoute](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定したルートテーブルに対してルートを作成します。ルートはすべてのトラフィックと一致し、これを指定されたインターネットゲートウェイに送信します。

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

出力:

```
True
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CreateRoute](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **CreateRouteTable**で を使用する

次のサンプルコードは、CreateRouteTable を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [プライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成する](#)
- [Amazon VPC の使用を開始する](#)

CLI

AWS CLI

ルートテーブルを作成するには

この例では、指定された VPC に対してルートテーブルを作成しています。

コマンド:

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

出力:

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

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateRouteTable](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定された VPC に対してルートテーブルを作成します。

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

出力:

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CreateRouteTable](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定された VPC に対してルートテーブルを作成します。

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

出力:

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CreateRouteTable](#)」を参照してください。

Ruby

SDK for Ruby

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

```

# Prerequisites:
#
# - A VPC in Amazon VPC.
# - A subnet in that VPC.
# - A gateway attached to that subnet.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param vpc_id [String] The ID of the VPC for the route table.
# @param subnet_id [String] The ID of the subnet for the route table.
# @param gateway_id [String] The ID of the gateway for the route.
# @param destination_cidr_block [String] The destination CIDR block
#   for the route.
# @param tag_key [String] The key portion of the tag for the route table.
# @param tag_value [String] The value portion of the tag for the route table.
# @return [Boolean] true if the route table was created and associated;
#   otherwise, false.
# @example
#   exit 1 unless route_table_created_and_associated?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     'vpc-0b6f769731EXAMPLE',
#     'subnet-03d9303b57EXAMPLE',
#     'igw-06ca90c011EXAMPLE',
#     '0.0.0.0/0',
#     'my-key',
#     'my-value'
#   )
def route_table_created_and_associated?(
  ec2_resource,
  vpc_id,
  subnet_id,
  gateway_id,
  destination_cidr_block,
  tag_key,
  tag_value
)
  route_table = ec2_resource.create_route_table(vpc_id: vpc_id)
  puts "Created route table with ID '#{route_table.id}'."
  route_table.create_tags(
    tags: [
      {
        key: tag_key,
        value: tag_value

```

```

    }
  ]
)
puts 'Added tags to route table.'
route_table.create_route(
  destination_cidr_block: destination_cidr_block,
  gateway_id: gateway_id
)
puts 'Created route with destination CIDR block ' \
    "'#{destination_cidr_block}' and associated with gateway " \
    "'with ID '#{gateway_id}'."
route_table.associate_with_subnet(subnet_id: subnet_id)
puts "Associated route table with subnet with ID '#{subnet_id}'."
true
rescue StandardError => e
  puts "Error creating or associating route table: #{e.message}"
  puts 'If the route table was created but not associated, you should ' \
    'clean up by deleting the route table.'
  false
end

# Example usage:
def run_me
  vpc_id = ''
  subnet_id = ''
  gateway_id = ''
  destination_cidr_block = ''
  tag_key = ''
  tag_value = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage: ruby ec2-ruby-example-create-route-table.rb ' \
        'VPC_ID SUBNET_ID GATEWAY_ID DESTINATION_CIDR_BLOCK ' \
        'TAG_KEY TAG_VALUE REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-create-route-table.rb ' \
        'vpc-0b6f769731EXAMPLE subnet-03d9303b57EXAMPLE igw-06ca90c011EXAMPLE ' \
        "'0.0.0.0/0' my-key my-value us-west-2"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    vpc_id = 'vpc-0b6f769731EXAMPLE'
    subnet_id = 'subnet-03d9303b57EXAMPLE'

```

```
gateway_id = 'igw-06ca90c011EXAMPLE'
destination_cidr_block = '0.0.0.0/0'
tag_key = 'my-key'
tag_value = 'my-value'
# Replace us-west-2 with the AWS Region you're using for Amazon EC2.
region = 'us-west-2'
# Otherwise, use the values as specified at the command prompt.
else
  vpc_id = ARGV[0]
  subnet_id = ARGV[1]
  gateway_id = ARGV[2]
  destination_cidr_block = ARGV[3]
  tag_key = ARGV[4]
  tag_value = ARGV[5]
  region = ARGV[6]
end

ec2_resource = Aws::EC2::Resource.new(region: region)

if route_table_created_and_associated?(
  ec2_resource,
  vpc_id,
  subnet_id,
  gateway_id,
  destination_cidr_block,
  tag_key,
  tag_value
)
  puts 'Route table created and associated.'
else
  puts 'Route table not created or not associated.'
end
end

run_me if $PROGRAM_NAME == __FILE__
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[CreateRouteTable](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI `CreateSecurityGroup`で を使用する

次のサンプルコードは、`CreateSecurityGroup` を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)
- [プライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成する](#)
- [Amazon VPC の使用を開始する](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

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

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

- APIの詳細については、AWS SDK for .NET API リファレンスの「[CreateSecurityGroup](#)」を参照してください。

Bash

AWS CLI Bash スクリプトを使用する

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
#####
# 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

```

```
}
```

この例で使用されているユーティリティ関数。

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

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateSecurityGroup](#)」を参照してください。

C++

SDK for C++

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
//! Create a security group.
/*!
    \param groupName: A security group name.
    \param description: A description.
    \param vpcID: A virtual private cloud (VPC) ID.
    \param[out] groupIDResult: A string to receive the group ID.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::EC2::createSecurityGroup(const Aws::String &groupName,
                                     const Aws::String &description,
                                     const Aws::String &vpcID,
                                     Aws::String &groupIDResult,
                                     const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::CreateSecurityGroupRequest request;

    request.SetGroupName(groupName);
    request.SetDescription(description);
    request.SetVpcId(vpcID);
```

```
const Aws::EC2::Model::CreateSecurityGroupOutcome outcome =
    ec2Client.CreateSecurityGroup(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Failed to create security group:" <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}

std::cout << "Successfully created security group named " << groupName <<
    std::endl;

groupIDResult = outcome.GetResult().GetGroupId();

return true;
}
```

- APIの詳細については、AWS SDK for C++ API リファレンスの「[CreateSecurityGroup](#)」を参照してください。

CLI

AWS CLI

EC2-Classical 用セキュリティグループを作成するには

この例では、MySecurityGroup という名前のセキュリティグループが作成されます。

コマンド:

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

出力:

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

EC2-VPC 用セキュリティグループを作成するには

この例では、指定された VPC 用に MySecurityGroup という名前のセキュリティグループが作成されます。

コマンド:

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

出力:

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

詳細については、「AWS コマンドラインインターフェイスユーザーガイド」でセキュリティグループの使用方法を参照してください。

- API の詳細については、「AWS CLI Command Reference」の「[CreateSecurityGroup](#)」を参照してください。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/**
 * Creates a new security group asynchronously with the specified group name,
 * description, and VPC ID. It also
 *   * authorizes inbound traffic on ports 80 and 22 from the specified IP
 * address.
 *
 * @param groupName    the name of the security group to create
 * @param groupDesc    the description of the security group
```

```

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

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

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

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

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

```

```

        .build();

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

```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[CreateSecurityGroup](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

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

/**
 * Creates a security group.
 * @param {{ groupName: string, description: string }} options
 */
export const main = async ({ groupName, description }) => {
    const client = new EC2Client({});

```



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

try {
  const { GroupId } = await client.send(command);
  console.log(GroupId);
} catch (caught) {
  if (caught instanceof Error && caught.name === "InvalidParameterValue") {
    console.warn(`${caught.message}.`);
  } else {
    throw caught;
  }
}
};
```

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の「[CreateSecurityGroup](#)」を参照してください。

Kotlin

SDK for Kotlin

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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
    }
}
```

- APIの詳細については、「AWS SDK for Kotlin API リファレンス」の「[CreateSecurityGroup](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定された VPC 用にセキュリティグループを作成します。

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

出力:

```
sg-12345678
```

例 2: この例では、EC2-Classic のセキュリティグループを作成します。

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

出力:

```
sg-45678901
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CreateSecurityGroup](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定された VPC 用にセキュリティグループを作成します。

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

出力:

```
sg-12345678
```

例 2: この例では、EC2-Classic のセキュリティグループを作成します。

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

出力:

```
sg-45678901
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CreateSecurityGroup](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
        security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a
high-level identifier
                               that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.
```

```

        :return: An instance of SecurityGroupWrapper initialized with the default
        EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def create(self, group_name: str, group_description: str) -> str:
        """
        Creates a security group in the default virtual private cloud (VPC) of
        the current account.

        :param group_name: The name of the security group to create.
        :param group_description: The description of the security group to
        create.
        :return: The ID of the newly created security group.
        :raise Handles AWS SDK service-level ClientError, with special handling
        for ResourceAlreadyExists
        """
        try:
            response = self.ec2_client.create_security_group(
                GroupName=group_name, Description=group_description
            )
            self.security_group = response["GroupId"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExists":
                logger.error(
                    f"Security group '{group_name}' already exists. Please choose
                    a different name."
                )
                raise
            else:
                return self.security_group

```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[CreateSecurityGroup](#)」を参照してください。

Ruby

SDK for Ruby

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
# This code example does the following:
# 1. Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
# 2. Adds inbound rules to the security group.
# 3. Displays information about available security groups.
# 4. Deletes the security group.

require 'aws-sdk-ec2'

# Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
#
# Prerequisites:
#
# - A VPC in Amazon Virtual Private Cloud (Amazon VPC).
#
# @param ec2_client [Aws::EC2::Client] An initialized
#   Amazon EC2 client.
# @param group_name [String] A name for the security group.
# @param description [String] A description for the security group.
# @param vpc_id [String] The ID of the VPC for the security group.
# @return [String] The ID of security group that was created.
# @example
#   puts create_security_group(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-security-group',
#     'This is my security group.',
#     'vpc-6713dfEX'
#   )
def create_security_group(ec2_client, group_name, description, vpc_id)
  security_group = ec2_client.create_security_group(
    group_name: group_name,
    description: description,
```

```

    vpc_id: vpc_id
  )
  puts "Created security group '#{group_name}' with ID " \
    "'#{security_group.group_id}' in VPC with ID '#{vpc_id}'."
  security_group.group_id
rescue StandardError => e
  puts "Error creating security group: #{e.message}"
  'Error'
end

# Adds an inbound rule to an Amazon Elastic Compute Cloud (Amazon EC2)
# security group.
#
# Prerequisites:
#
# - The security group.
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @param security_group_id [String] The ID of the security group.
# @param ip_protocol [String] The network protocol for the inbound rule.
# @param from_port [String] The originating port for the inbound rule.
# @param to_port [String] The destination port for the inbound rule.
# @param cidr_ip_range [String] The CIDR IP range for the inbound rule.
# @return
# @example
#   exit 1 unless security_group_ingress_authorized?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'sg-030a858e078f1b9EX',
#     'tcp',
#     '80',
#     '80',
#     '0.0.0.0/0'
#   )
def security_group_ingress_authorized?(
  ec2_client, security_group_id, ip_protocol, from_port, to_port, cidr_ip_range
)
  ec2_client.authorize_security_group_ingress(
    group_id: security_group_id,
    ip_permissions: [
      {
        ip_protocol: ip_protocol,
        from_port: from_port,
        to_port: to_port,
        ip_ranges: [

```

```

        {
            cidr_ip: cidr_ip_range
        }
    ]
}
]
)
puts "Added inbound rule to security group '#{security_group_id}' for protocol
" \
    "'#{ip_protocol}' from port '#{from_port}' to port '#{to_port}' " \
    "with CIDR IP range '#{cidr_ip_range}'."
true
rescue StandardError => e
    puts "Error adding inbound rule to security group: #{e.message}"
    false
end

# Refactored method to simplify complexity for describing security group
permissions
def format_port_information(perm)
    from_port_str = perm.from_port == '-1' || perm.from_port == -1 ? 'All' :
perm.from_port.to_s
    to_port_str = perm.to_port == '-1' || perm.to_port == -1 ? 'All' :
perm.to_port.to_s
    { from_port: from_port_str, to_port: to_port_str }
end

# Displays information about a security group's IP permissions set in
# Amazon Elastic Compute Cloud (Amazon EC2).
def describe_security_group_permissions(perm)
    ports = format_port_information(perm)

    print "  Protocol: #{perm.ip_protocol == '-1' ? 'All' : perm.ip_protocol}"
    print ", From: #{ports[:from_port]}, To: #{ports[:to_port]}"

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

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

# Displays information about available security groups in

```



```
# Amazon Elastic Compute Cloud (Amazon EC2).
def describe_security_groups(ec2_client)
  response = ec2_client.describe_security_groups

  if response.security_groups.count.positive?
    response.security_groups.each do |sg|
      display_group_details(sg)
    end
  else
    puts 'No security groups found.'
  end
rescue StandardError => e
  puts "Error getting information about security groups: #{e.message}"
end

# Helper method to display the details of security groups
def display_group_details(sg)
  puts '-' * (sg.group_name.length + 13)
  puts "Name:          #{sg.group_name}"
  puts "Description:   #{sg.description}"
  puts "Group ID:      #{sg.group_id}"
  puts "Owner ID:      #{sg.owner_id}"
  puts "VPC ID:        #{sg.vpc_id}"

  display_group_tags(sg.tags) if sg.tags.count.positive?
  display_group_permissions(sg)
end

def display_group_tags(tags)
  puts 'Tags:'
  tags.each do |tag|
    puts "  Key: #{tag.key}, Value: #{tag.value}"
  end
end

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__
```

- API の詳細については、AWS SDK for Ruby API リファレンスの「[CreateSecurityGroup](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn create_security_group(
    &self,
    name: &str,
    description: &str,
) -> Result<SecurityGroup, EC2Error> {
    tracing::info!("Creating security group {name}");
    let create_output = self
        .client
        .create_security_group()
        .group_name(name)
        .description(description)
        .send()
        .await
        .map_err(EC2Error::from)?;

    let group_id = create_output
```

```

        .group_id
        .ok_or_else(|| EC2Error::new("Missing security group id after
creation"))?);

    let group = self
        .describe_security_group(&group_id)
        .await?
        .ok_or_else(|| {
            EC2Error::new(format!("Could not find security group with id
{group_id}"))
        })?;

    tracing::info!("Created security group {name} as {group_id}");

    Ok(group)
}

```

- APIの詳細については、「AWS SDK for Rust API リファレンス」の「[CreateSecurityGroup](#)」を参照してください。

SAP ABAP

SDK for SAP ABAP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

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.
```

- APIの詳細については、「AWS SDK for SAP ABAP API リファレンス」の「[CreateSecurityGroup](#)」を参照してください。

Swift

SDK for Swift

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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  
    }  
}
```

```
}  
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[CreateSecurityGroup](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **CreateSnapshot** を使用する

次のサンプルコードは、CreateSnapshot を使用する方法を説明しています。

CLI

AWS CLI

スナップショットを作成するには

このコマンド例では、vol-1234567890abcdef0 というボリューム ID を持つスナップショットと、そのスナップショットを識別するための短い説明を作成します。

コマンド:

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

出力:

```
{  
  "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"  
}
```

タグ付きのスナップショットを作成するには

このコマンド例では、スナップショットを作成し、purpose=prod と costcenter=123 という 2 つのタグを適用します。

コマンド:

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

出力:

```
{  
  "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"  
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateSnapshot](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたボリュームのスナップショットを作成します。

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

出力:

```
DataEncryptionKeyId :  
Description          : This is a test  
Encrypted            : False  
KmsKeyId             :  
OwnerAlias           :  
OwnerId              : 123456789012  
Progress             :  
SnapshotId           : snap-12345678  
StartTime            : 12/22/2015 1:28:42 AM  
State                : pending  
StateMessage         :  
Tags                 : {}  
VolumeId             : vol-12345678  
VolumeSize           : 20
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CreateSnapshot](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたボリュームのスナップショットを作成します。

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

出力:

```
DataEncryptionKeyId :  
Description          : This is a test  
Encrypted            : False  
KmsKeyId             :  
OwnerAlias           :
```

```
OwnerId           : 123456789012
Progress          :
SnapshotId        : snap-12345678
StartTime         : 12/22/2015 1:28:42 AM
State             : pending
StateMessage      :
Tags              : {}
VolumeId          : vol-12345678
VolumeSize        : 20
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CreateSnapshot](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **CreateSpotDatafeedSubscription** を使用する

次のサンプルコードは、CreateSpotDatafeedSubscription を使用する方法を説明しています。

CLI

AWS CLI

スポットインスタンスのデータフィードを作成するには

次の create-spot-datafeed-subscription の例では、スポットインスタンスのデータフィードを作成します。

```
aws ec2 create-spot-datafeed-subscription \
  --bucket amzn-s3-demo-bucket \
  --prefix spot-data-feed
```

出力:

```
{
  "SpotDatafeedSubscription": {
    "Bucket": "amzn-s3-demo-bucket",
```

```
"OwnerId": "123456789012",  
"Prefix": "spot-data-feed",  
"State": "Active"  
}  
}
```

データフィードは、指定した Amazon S3 バケットに保存されます。このデータフィードのファイル名の形式は次のとおりです。

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

詳細については、Amazon EC2 ユーザーガイドの「[スポットインスタンスのデータフィード](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateSpotDatafeedSubscription](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、スポットインスタンスのデータフィードを作成します。

```
New-EC2SpotDatafeedSubscription -Bucket amzn-s3-demo-bucket -Prefix spotdata
```

出力:

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CreateSpotDatafeedSubscription](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、スポットインスタンスのデータフィードを作成します。

```
New-EC2SpotDatafeedSubscription -Bucket amzn-s3-demo-bucket -Prefix spotdata
```

出力:

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CreateSpotDatafeedSubscription](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **CreateSubnet**で を使用する

次のサンプルコードは、CreateSubnet を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [プライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成する](#)
- [Amazon VPC の使用を開始する](#)
- [Transit Gateway を開始する](#)

CLI

AWS CLI

例 1: IPv4 CIDR ブロックのみを使用してサブネットを作成するには

次の create-subnet の例では、指定された IPv4 CIDR ブロックで指定された VPC にサブネットを作成しています。

```
aws ec2 create-subnet \  
  --vpc-id vpc-081ec835f3EXAMPLE \  
  --cidr-block 10.0.0.0/24
```

```
--cidr-block 10.0.0.0/24 \  
--tag-specifications ResourceType=subnet,Tags=[{Key=Name,Value=my-ipv4-only-  
subnet}]
```

出力:

```
{  
  "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"  
  }  
}
```

例 2: IPv4 と IPv6 CIDR ブロックの両方を使用してサブネットを作成するには

次の create-subnet の例では、指定された IPv4 および IPv6 CIDR ブロックで指定された VPC にサブネットを作成しています。

```
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}]
```

出力:

```
{
  "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"
  }
}
```

例 3: IPv6 CIDR ブロックのみを使用してサブネットを作成するには

次の `create-subnet` の例では、指定された IPv6 CIDR ブロックで指定された VPC にサブネットを作成しています。

```
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}]
```

出力:

```
{
  "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"
  }
}
```

詳細については、「Amazon VPC ユーザーガイド」の「[VPC とサブネット](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateSubnet](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定した CIDR でサブネットを作成します。

```
New-EC2Subnet -VpcId vpc-12345678 -CidrBlock 10.0.0.0/24
```

出力:

```
AvailabilityZone      : us-west-2c
AvailableIpAddressCount : 251
CidrBlock             : 10.0.0.0/24
DefaultForAz          : False
MapPublicIpOnLaunch   : False
State                 : pending
SubnetId              : subnet-1a2b3c4d
Tag                   : {}
VpcId                 : vpc-12345678
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CreateSubnet](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定した CIDR でサブネットを作成します。

```
New-EC2Subnet -VpcId vpc-12345678 -CidrBlock 10.0.0.0/24
```

出力:

```
AvailabilityZone      : us-west-2c
AvailableIpAddressCount : 251
CidrBlock             : 10.0.0.0/24
DefaultForAz          : False
MapPublicIpOnLaunch   : False
State                 : pending
SubnetId              : subnet-1a2b3c4d
Tag                   : {}
VpcId                 : vpc-12345678
```


- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CreateSubnet](#)」を参照してください。

Ruby

SDK for Ruby

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
require 'aws-sdk-ec2'

# Creates a subnet within a virtual private cloud (VPC) in
# Amazon Virtual Private Cloud (Amazon VPC) and then tags
# the subnet.
#
# Prerequisites:
#
# - A VPC in Amazon VPC.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param vpc_id [String] The ID of the VPC for the subnet.
# @param cidr_block [String] The IPv4 CIDR block for the subnet.
# @param availability_zone [String] The ID of the Availability Zone
#   for the subnet.
# @param tag_key [String] The key portion of the tag for the subnet.
# @param tag_value [String] The value portion of the tag for the subnet.
# @return [Boolean] true if the subnet was created and tagged;
#   otherwise, false.
# @example
#   exit 1 unless subnet_created_and_tagged?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     'vpc-6713dfEX',
#     '10.0.0.0/24',
#     'us-west-2a',
#     'my-key',
```

```
# 'my-value'
# )
def subnet_created_and_tagged?(
  ec2_resource,
  vpc_id,
  cidr_block,
  availability_zone,
  tag_key,
  tag_value
)
  subnet = ec2_resource.create_subnet(
    vpc_id: vpc_id,
    cidr_block: cidr_block,
    availability_zone: availability_zone
  )
  subnet.create_tags(
    tags: [
      {
        key: tag_key,
        value: tag_value
      }
    ]
  )
  puts "Subnet created with ID '#{subnet.id}' in VPC with ID '#{vpc_id}' " \
    "and CIDR block '#{cidr_block}' in availability zone " \
    "'#{availability_zone}' and tagged with key '#{tag_key}' and " \
    "value '#{tag_value}'."
  true
rescue StandardError => e
  puts "Error creating or tagging subnet: #{e.message}"
  false
end

# Example usage:
def run_me
  vpc_id = ''
  cidr_block = ''
  availability_zone = ''
  tag_key = ''
  tag_value = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:  ruby ec2-ruby-example-create-subnet.rb ' \
```

```
'VPC_ID CIDR_BLOCK AVAILABILITY_ZONE TAG_KEY TAG_VALUE REGION'
# Replace us-west-2 with the AWS Region you're using for Amazon EC2.
puts 'Example: ruby ec2-ruby-example-create-subnet.rb ' \
    'vpc-6713dfEX 10.0.0.0/24 us-west-2a my-key my-value us-west-2'
exit 1
# If no values are specified at the command prompt, use these default values.
elsif ARGV.count.zero?
  vpc_id = 'vpc-6713dfEX'
  cidr_block = '10.0.0.0/24'
  availability_zone = 'us-west-2a'
  tag_key = 'my-key'
  tag_value = 'my-value'
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  region = 'us-west-2'
# Otherwise, use the values as specified at the command prompt.
else
  vpc_id = ARGV[0]
  cidr_block = ARGV[1]
  availability_zone = ARGV[2]
  tag_key = ARGV[3]
  tag_value = ARGV[4]
  region = ARGV[5]
end

ec2_resource = Aws::EC2::Resource.new(region: region)

if subnet_created_and_tagged?(
  ec2_resource,
  vpc_id,
  cidr_block,
  availability_zone,
  tag_key,
  tag_value
)
  puts 'Subnet created and tagged.'
else
  puts 'Subnet not created or not tagged.'
end
end

run_me if $PROGRAM_NAME == __FILE__
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[CreateSubnet](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **CreateTags**で を使用する

次のサンプルコードは、CreateTags を使用する方法を説明しています。

C++

SDK for C++

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
//! Add or overwrite only the specified tags for the specified Amazon Elastic
Compute Cloud (Amazon EC2) resource or resources.
/*!
    \param resources: The resources for the tags.
    \param tags: Vector of tags.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::EC2::createTags(const Aws::Vector<Aws::String> &resources,
                             const Aws::Vector<Aws::EC2::Model::Tag> &tags,
                             const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::CreateTagsRequest createTagsRequest;
    createTagsRequest.SetResources(resources);
    createTagsRequest.SetTags(tags);

    Aws::EC2::Model::CreateTagsOutcome outcome =
    ec2Client.CreateTags(createTagsRequest);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created tags for resources" << std::endl;
    } else {
        std::cerr << "Failed to create tags for resources, " <<
outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「[CreateTags](#)」を参照してください。

CLI

AWS CLI

例 1: リソースにタグを追加するには

次の create-tags の例では、タグ Stack=production を指定されたイメージに追加するか、タグキーが Stack の AMI 用に既存のタグを上書きします。

```
aws ec2 create-tags \
  --resources ami-1234567890abcdef0 \
  --tags Key=Stack,Value=production
```

このコマンドでは何も出力されません。

例 2: 複数のリソースにタグを追加するには

次の create-tags の例では、2 つのタグを AMI とインスタンス用に追加 (または上書き) します。一方のタグでは、キー (webserver) はありますが値はありません (値は空文字列に設定されています)。もう一方のタグにはキー (stack) と値 (Production) があります。

```
aws ec2 create-tags \
  --resources ami-1a2b3c4d i-1234567890abcdef0 \
  --tags Key=webserver,Value= Key=stack,Value=Production
```

このコマンドでは何も出力されません。

例 3: 特殊文字を含むタグを追加するには

次の `create-tags` の例では、インスタンスにタグ `[Group]=test` を追加します。角括弧 (`[`, `]`) は特殊文字であり、エスケープする必要があります。以下の例でも、各環境に適した行継続文字を使用しています。

Windows を使用している場合、特殊文字を含む要素を二重引用符 (`"`) で囲み、各二重引用符の前にバックスラッシュ (`\`) を付けます。

```
aws ec2 create-tags ^
  --resources i-1234567890abcdef0 ^
  --tags Key=\"[Group]\",Value=test
```

Windows PowerShell を使用している場合、次のように特殊文字を含む値を二重引用符 (`"`) で囲み、各二重引用符の前にバックスラッシュ (`\`) を付けてから、キーと値の構造全体を一重引用符 (`'`) で囲みます。

```
aws ec2 create-tags `
  --resources i-1234567890abcdef0 `
  --tags 'Key=\"[Group]\",Value=test'
```

Linux または OS X を使用している場合は、次のように特殊文字を含む要素を二重引用符 (`"`) で囲んだ後、キーと値の構造全体を一重引用符 (`'`) で囲みます。

```
aws ec2 create-tags \
  --resources i-1234567890abcdef0 \
  --tags 'Key="[Group]",Value=test'
```

詳細については、「Amazon EC2 ユーザーガイド」の「[Amazon EC2 リソースのタグ付け](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateTags](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定したリソースに単一のタグを追加します。タグキーは「myTag」で、タグ値は「myTagValue」です。この例で使用される構文には、PowerShell バージョン 3 以降が必要です。

```
New-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag"; Value="myTagValue" }
```

例 2: この例では、指定されたリソースに指定されたタグを更新または追加します。この例で使用される構文には、PowerShell バージョン 3 以降が必要です。

```
New-EC2Tag -Resource i-12345678 -Tag @( @{ Key="myTag"; Value="newTagValue" },  
    @{ Key="test"; Value="anotherTagValue" } )
```

例 3: PowerShell バージョン 2 では、New-Object を使用してタグパラメータのタグを作成する必要があります。

```
$tag = New-Object Amazon.EC2.Model.Tag  
$tag.Key = "myTag"  
$tag.Value = "myTagValue"  
  
New-EC2Tag -Resource i-12345678 -Tag $tag
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CreateTags](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定したリソースに単一のタグを追加します。タグキーは「myTag」で、タグ値は「myTagValue」です。この例で使用される構文には、PowerShell バージョン 3 以降が必要です。

```
New-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag"; Value="myTagValue" }
```

例 2: この例では、指定されたリソースに指定されたタグを更新または追加します。この例で使用される構文には、PowerShell バージョン 3 以降が必要です。

```
New-EC2Tag -Resource i-12345678 -Tag @( @{ Key="myTag"; Value="newTagValue" },  
    @{ Key="test"; Value="anotherTagValue" } )
```

例 3: PowerShell バージョン 2 では、New-Object を使用してタグパラメータのタグを作成する必要があります。

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

New-EC2Tag -Resource i-12345678 -Tag $tag
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CreateTags](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

この例では、インスタンスの作成後に名前タグを適用します。

```
pub async fn create_instance<'a>(
    &self,
    image_id: &'a str,
    instance_type: InstanceType,
    key_pair: &'a KeyPairInfo,
    security_groups: Vec<&'a SecurityGroup>,
) -> Result<String, EC2Error> {
    let run_instances = self
        .client
        .run_instances()
        .image_id(image_id)
        .instance_type(instance_type)
        .key_name(
            key_pair
                .key_name()
                .ok_or_else(|| EC2Error::new("Missing key name when launching
instance"))?,
```



```
)
.set_security_group_ids(Some(
  security_groups
    .iter()
    .filter_map(|sg| sg.group_id.clone())
    .collect(),
))
.min_count(1)
.max_count(1)
.send()
.await?;

if run_instances.instances().is_empty() {
  return Err(EC2Error::new("Failed to create instance"));
}

let instance_id = run_instances.instances()[0].instance_id().unwrap();
let response = self
  .client
  .create_tags()
  .resources(instance_id)
  .tags(
    Tag::builder()
      .key("Name")
      .value("From SDK Examples")
      .build(),
  )
  .send()
  .await;

match response {
  Ok(_) => tracing::info!("Created {instance_id} and applied tags."),
  Err(err) => {
    tracing::info!("Error applying tags to {instance_id}: {err:?}");
    return Err(err.into());
  }
}

tracing::info!("Instance is created.");

Ok(instance_id.to_string())
}
```

- API の詳細については、AWS SDK for Rust API リファレンスの「[CreateTags](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **CreateVolume** を使用する

次のサンプルコードは、CreateVolume を使用する方法を説明しています。

CLI

AWS CLI

空の汎用 SSD (gp2) ボリュームを作成するには

次の create-volume の例では、指定したアベイラビリティーゾーンに 80 GiB 汎用 SSD (gp2) ボリュームを作成します。現在のリージョンは us-east-1 である必要があります。または、--region パラメータを追加してコマンドのリージョンを指定できます。

```
aws ec2 create-volume \  
  --volume-type gp2 \  
  --size 80 \  
  --availability-zone us-east-1a
```

出力:

```
{  
  "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  
}
```

ボリュームタイプを指定しない場合、デフォルトのボリュームタイプは gp2 です。

```
aws ec2 create-volume \  
  --size 80 \  
  --availability-zone us-east-1a
```

例 2: スナップショットからプロビジョンド IOPS SSD (io1) ボリュームを作成するには

次の create-volume の例では、指定したスナップショットを使用して、指定したアベイラビリティゾーンに 1,000 個のプロビジョニングされた IOPS を持つプロビジョンド IOPS SSD (io1) ボリュームを作成します。

```
aws ec2 create-volume \  
  --volume-type io1 \  
  --iops 1000 \  
  --snapshot-id snap-066877671789bd71b \  
  --availability-zone us-east-1a
```

出力:

```
{  
  "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  
}
```

例 3: 暗号化されたボリュームを作成するには

次の create-volume の例では、EBS 暗号化用のデフォルトの CMK を使用して暗号化されたボリュームを作成します。デフォルトで暗号化が無効になっている場合、次のように --encrypted パラメータを指定する必要があります。

```
aws ec2 create-volume \  
  --size 80 \  
  --encrypted
```

```
--encrypted \  
--availability-zone us-east-1a
```

出力:

```
{  
  "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  
}
```

暗号化がデフォルトで有効になっている場合、次の例のコマンドは、`--encrypted` パラメータがなくても暗号化されたボリュームを作成します。

```
aws ec2 create-volume \  
  --size 80 \  
  --availability-zone us-east-1a
```

`--kms-key-id` パラメータを使用してカスタマー管理の CMK を指定する場合は、デフォルトで暗号化が有効になっていても `--encrypted` パラメータを指定する必要があります。

```
aws ec2 create-volume \  
  --volume-type gp2 \  
  --size 80 \  
  --encrypted \  
  --kms-key-id 0ea3fef3-80a7-4778-9d8c-1c0c6EXAMPLE \  
  --availability-zone us-east-1a
```

例 4: タグ付きのボリュームを作成するには

次の `create-volume` の例では、ボリュームを作成して 2 つのタグを追加します。

```
aws ec2 create-volume \  
  --availability-zone us-east-1a \  
  --volume-type gp2 \  
  --tags '[{"Key": "Name", "Value": "MyVolume"}, {"Key": "Environment", "Value": "Production"}]'
```

```
--size 80 \  
--tag-specifications  
'ResourceType=volume,Tags=[{Key=purpose,Value=production},{Key=cost-  
center,Value=cc123}]'
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateVolume](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定したボリュームを作成します。

```
New-EC2Volume -Size 50 -AvailabilityZone us-west-2a -VolumeType gp2
```

出力:

```
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
```

例 2: このリクエスト例では、ボリュームを作成し、キーがスタック、値が本番環境のタグを適用します。

```
$tag = @{ Key="stack"; Value="production" }  
  
$tagspec = new-object Amazon.EC2.Model.TagSpecification  
$tagspec.ResourceType = "volume"  
$tagspec.Tags.Add($tag)  
  
New-EC2Volume -Size 80 -AvailabilityZone "us-west-2a" -TagSpecification $tagspec
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CreateVolume](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定したボリュームを作成します。

```
New-EC2Volume -Size 50 -AvailabilityZone us-west-2a -VolumeType gp2
```

出力:

```
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
```

例 2: このリクエスト例では、ボリュームを作成し、キーがスタック、値が本番環境のタグを適用します。

```
$tag = @{ Key="stack"; Value="production" }

$tagspec = new-object Amazon.EC2.Model.TagSpecification
$tagspec.ResourceType = "volume"
$tagspec.Tags.Add($tag)

New-EC2Volume -Size 80 -AvailabilityZone "us-west-2a" -TagSpecification $tagspec
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CreateVolume](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **CreateVpc**で を使用する

次のサンプルコードは、CreateVpc を使用方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [プライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成する](#)
- [Amazon VPC の使用を開始する](#)
- [Transit Gateway を開始する](#)
- [VPC IPAM の開始方法](#)

CLI

AWS CLI

例 1: VPC を作成するには

次の create-vpc の例では、指定された IPv4 CIDR ブロックと Name タグを持つ VPC を作成しています。

```
aws ec2 create-vpc \  
  --cidr-block 10.0.0.0/16 \  
  --tag-specifications ResourceType=vpc, Tags=[{Key=Name, Value=MyVpc}]
```

出力:

```
{  
  "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"
    }
  ]
}
}

```

例 2: 専用テナンシーを持つ VPC を作成するには

次の `create-vpc` の例では、指定された IPv4 CIDR ブロックと専用テナンシーを持つ VPC を作成しています。

```

aws ec2 create-vpc \
  --cidr-block 10.0.0.0/16 \
  --instance-tenancy dedicated

```

出力:

```

{
  "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
  }
}
```

例 3: IPv6 CIDR ブロックで VPC を作成するには

次の `create-vpc` の例では、Amazon が提供する IPv6 CIDR ブロックを使用して、VPC を作成しています。

```
aws ec2 create-vpc \
  --cidr-block 10.0.0.0/16 \
  --amazon-provided-ipv6-cidr-block
```

出力:

```
{
  "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
  }
}

```

例 4: IPAM プールの CIDR を使って VPC を作成するには

次の create-vpc の例では、Amazon VPC IP Address Manager (IPAM) プールの CIDR を使用して VPC を作成しています。

Linux および 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"}]

```

出力:

```

{
  "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"
    }
  ]
}
}

```

詳細については、「Amazon VPC IPAM ユーザーガイド」の「[IPAM プール CIDR を使用する VPC を作成する](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateVpc](#)」を参照してください。

PHP

SDK for PHP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

/**
 * @param string $cidr
 * @return array
 */
public function createVpc(string $cidr): array
{
    try {
        $result = $this->ec2Client->createVpc([
            "CidrBlock" => $cidr,

```

```
    });  
    return $result['Vpc'];  
} catch (Ec2Exception $caught){  
    echo "There was a problem creating the VPC: {$caught->getAwsErrorMessage()}\n";  
    throw $caught;  
}  
}
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「[CreateVpc](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定した CIDR で VPC を作成します。Amazon VPC では、VPC 用に、デフォルトの DHCP オプションセット、メインルートテーブル、デフォルトのネットワーク ACL も作成します。

```
New-EC2VPC -CidrBlock 10.0.0.0/16
```

出力:

```
CidrBlock      : 10.0.0.0/16  
DhcpOptionsId  : dopt-1a2b3c4d  
InstanceTenancy : default  
IsDefault      : False  
State          : pending  
Tags           : {}  
VpcId          : vpc-12345678
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CreateVpc](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定した CIDR で VPC を作成します。Amazon VPC では、VPC 用に、デフォルトの DHCP オプションセット、メインルートテーブル、デフォルトのネットワーク ACL も作成します。

```
New-EC2VPC -CidrBlock 10.0.0.0/16
```

出力:

```
CidrBlock      : 10.0.0.0/16
DhcpOptionsId  : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault      : False
State          : pending
Tags           : {}
VpcId          : vpc-12345678
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CreateVpc](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[CreateVpc](#)」を参照してください。

Ruby

SDK for Ruby

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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 })
end
```

```

vpc.create_tags(tags: [{ key: tag_key, value: tag_value }])

puts "Created VPC with ID '#{vpc.id}' and tagged with key " \
    "'#{tag_key}' and value '#{tag_value}'."
true
rescue StandardError => e
  puts e.message
  false
end

# Example usage:
def run_me
  cidr_block = ''
  tag_key = ''
  tag_value = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:  ruby ec2-ruby-example-create-vpc.rb ' \
        'CIDR_BLOCK TAG_KEY TAG_VALUE REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-create-vpc.rb ' \
        '10.0.0.0/24 my-key my-value us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    cidr_block = '10.0.0.0/24'
    tag_key = 'my-key'
    tag_value = 'my-value'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    cidr_block = ARGV[0]
    tag_key = ARGV[1]
    tag_value = ARGV[2]
    region = ARGV[3]
  end

  ec2_resource = Aws::EC2::Resource.new(region: region)

  if vpc_created_and_tagged?(
    ec2_resource,

```



```

    cidr_block,
    tag_key,
    tag_value
  )
  puts 'VPC created and tagged.'
else
  puts 'VPC not created or not tagged.'
end
end
end

run_me if $PROGRAM_NAME == __FILE__

```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[CreateVpc](#)」を参照してください。

SAP ABAP

SDK for SAP ABAP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

" 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.

```

- API の詳細については、AWS SDK for SAP [CreateVpc](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI `CreateVpcEndpoint` で 使用する

次のサンプルコードは、`CreateVpcEndpoint` を使用方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [プライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成する](#)

CLI

AWS CLI

例 1: ゲートウェイエンドポイントを作成するには

次の `create-vpc-endpoint` の例では、`us-east-1` リージョン内の VPC `vpc-1a2b3c4d` と Amazon S3 の間にゲートウェイ VPC エンドポイントを作成し、ルートテーブル `rtb-11aa22bb` をそのエンドポイントに関連付けます。

```
aws ec2 create-vpc-endpoint \
  --vpc-id vpc-1a2b3c4d \
  --service-name com.amazonaws.us-east-1.s3 \
  --route-table-ids rtb-11aa22bb
```

出力:

```
{
  "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"
  }
}
```

詳細については、AWS PrivateLink ユーザーガイドの「[ゲートウェイエンドポイントを作成する](#)」を参照してください。

例 2: インターフェイスエンドポイントを作成するには

次の `create-vpc-endpoint` の例では、`us-east-1` リージョン内の VPC `vpc-1a2b3c4d` と Amazon S3 の間にインターフェイス VPC エンドポイントを作成します。コマンドにより、サブネット `subnet-1a2b3c4d` にエンドポイントが作成され、セキュリティグループ `sg-1a2b3c4d` に関連付けられ、「サービス」というキーと「S3」という値を持つタグが追加されます。

```
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}]
```

出力:

```
{
  "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"
}

```

詳細については、AWS PrivateLink ユーザーガイドの「[インターフェイス VPC エンドポイントを作成する](#)」を参照してください。

例 3: Gateway Load Balancer エンドポイントを作成するには

次の `create-vpc-endpoint` の例では、VPC `vpc-111122223333aabbc` と、Gateway Load Balancer を使用して設定されたサービスの間に Gateway Load Balancer エンドポイントを作成します。

```

aws ec2 create-vpc-endpoint \
  --service-name com.amazonaws.vpce.us-east-1.vpce-svc-123123a1c43abc123 \
  --vpc-endpoint-type GatewayLoadBalancer \
  --vpc-id vpc-111122223333aabbc \
  --subnet-ids subnet-0011aabbcc2233445

```

出力:

```
{
  "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"
  }
}
```

詳細については、AWS PrivateLink ユーザーガイドの「[Gateway Load Balancer エンドポイント](#)」を参照してください。

例 4: リソースエンドポイントを作成するには

次の create-vpc-endpoint の例では、リソースエンドポイントを作成します。

```
aws ec2 create-vpc-endpoint \
  --vpc-endpoint-type Resource \
  --vpc-id vpc-111122223333aabbcc \
  --subnet-ids subnet-0011aabbcc2233445 \
  --resource-configuration-arn arn:aws:vpce-us-east-1:123456789012:resourceconfiguration/rcfg-0123abcde98765432
```

出力:

```
{
  "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"
  }
}

```

詳細については、「AWS PrivateLink ユーザーガイド」の「[Resource endpoints](#)」を参照してください。

例 5: サービスネットワークエンドポイントを作成するには

次の `create-vpc-endpoint` の例では、サービスネットワークエンドポイントを作成します。

```

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

```

出力:

```

{
  "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"
}

```

詳細については、「AWS PrivateLink ユーザーガイド」の「[Service network endpoints](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateVpcEndpoint](#)」を参照してください。

PHP

SDK for PHP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

/**
 * @param string $serviceName
 * @param string $vpcId
 * @param array $routeTableIds
 * @return array
 */

```

```

public function createVpcEndpoint(string $serviceName, string $vpcId, array
$routeTableIds): array
{
    try {
        $result = $this->ec2Client->createVpcEndpoint([
            'ServiceName' => $serviceName,
            'VpcId' => $vpcId,
            'RouteTableIds' => $routeTableIds,
        ]);

        return $result["VpcEndpoint"];
    } catch (Ec2Exception $caught){
        echo "There was a problem creating the VPC Endpoint: {$caught-
>getAwsErrorMessage()}\n";
        throw $caught;
    }
}

```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「[CreateVpcEndpoint](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、VPC vpc-0fc1ff23f45b678eb でサービス com.amazonaws.eu-west-1.s3 の新しい VPC エンドポイントを作成します。

```

New-EC2VpcEndpoint -ServiceName com.amazonaws.eu-west-1.s3 -VpcId
vpc-0fc1ff23f45b678eb

```

出力:

```

ClientToken VpcEndpoint
-----
Amazon.EC2.Model.VpcEndpoint

```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CreateVpcEndpoint](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、VPC `vpc-0fc1ff23f45b678eb` でサービス `com.amazonaws.eu-west-1.s3` の新しい VPC エンドポイントを作成します。

```
New-EC2VpcEndpoint -ServiceName com.amazonaws.eu-west-1.s3 -VpcId
vpc-0fc1ff23f45b678eb
```

出力:

```
ClientToken VpcEndpoint
-----
Amazon.EC2.Model.VpcEndpoint
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CreateVpcEndpoint](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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

```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[CreateVpcEndpoint](#)」を参照してください。

SAP ABAP

SDK for SAP ABAP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
" 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.
```

- API の詳細については、AWS SDK for SAP ABAP API リファレンスの「[CreateVpcEndpoint](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で `CreateVpnConnection` を使用する

次のサンプルコードは、`CreateVpnConnection` を使用方法を説明しています。

CLI

AWS CLI

例 1: 動的ルーティングを使用して VPN 接続を作成するには

次の `create-vpn-connection` の例では、指定した仮想プライベートゲートウェイと指定したカスタマーゲートウェイの間に VPN 接続を作成し、VPN 接続にタグを適用します。出力には、カスタマーゲートウェイデバイスの設定情報が XML 形式で含まれます。

```
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}]'
```

出力:

```
{
  "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": [
        {},
        {}
      ]
    },
    "Routes": [],
```

```

    "Tags": [
      {
        "Key": "Name",
        "Value": "BGP-VPN"
      }
    ]
  }
}

```

詳細については、[AWS 「 Site-to-Site VPN ユーザーガイドAWS 」 のSite-to-Site VPN works](#)」を参照してください。

例 2: 静的ルーティングを使用して VPN 接続を作成するには

次の create-vpn-connection の例では、指定した仮想プライベートゲートウェイと指定したカスタマーゲートウェイの間に VPN 接続を作成します。オプションで静的ルーティングを指定します。出力には、カスタマーゲートウェイデバイスの設定情報が XML 形式で含まれます。

```

aws ec2 create-vpn-connection \
  --type ipsec.1 \
  --customer-gateway-id cgw-001122334455aabbcc \
  --vpn-gateway-id vgw-1a1a1a1a1a1a2b2b2 \
  --options "{\"StaticRoutesOnly\":true}"

```

出力:

```

{
  "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": []
}
}

```

詳細については、[AWS 「Site-to-Site VPN ユーザーガイド」](#) の Site-to-Site VPN works」を参照してください。

例 3: VPN 接続を作成し、CIDR 内で独自のキーと事前共有キーを指定するには

次の create-vpn-connection の例では、VPN 接続を作成し、内部 IP アドレス CIDR ブロック、および各トンネルのカスタム事前共有キーを指定します。指定された値は CustomerGatewayConfiguration 情報に返されます。

```

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}]'

```

出力:

```

{
  "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": []
}
}

```

詳細については、[AWS 「 Site-to-Site VPN ユーザーガイドAWS 」 のSite-to-Site VPN works](#)」を参照してください。

例 4: IPv6 トラフィックをサポートする VPN 接続を作成するには

次の `create-vpn-connection` の例では、指定した Transit Gateway と指定したカスタマーゲートウェイの間に IPv6 トラフィックをサポートする VPN 接続を作成します。両方のトンネルのトンネルオプションでは、`g` IKE ネゴシエーションを開始 AWS する必要があることを指定します。

```

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}]

```

出力:

```

{
  "VpnConnection": {
    "CustomerGatewayConfiguration": "..configuration information...",
    "CustomerGatewayId": "cgw-001122334455aabbcc",
    "Category": "VPN",
    "State": "pending",
  }
}

```

```

    "VpnConnectionId": "vpn-1111111122222222",
    "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": []
}

```

詳細については、[AWS 「Site-to-Site VPN ユーザーガイド」](#) の「Site-to-Site VPN works」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateVpnConnection](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定した仮想プライベートゲートウェイと指定したカスタマーゲートウェイの間に VPN 接続を作成します。出力には、ネットワーク管理者が必要とする設定情報が XML 形式で含まれます。

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId vgw-1a2b3c4d
```

出力:


```
CustomerGatewayConfiguration : [XML document]
CustomerGatewayId           : cgw-1a2b3c4d
Options                     :
Routes                     : {}
State                      : pending
Tags                      : {}
Type                      :
VgwTelemetry               : {}
VpnConnectionId            : vpn-12345678
VpnGatewayId               : vgw-1a2b3c4d
```

例 2: この例では、VPN 接続を作成し、指定された名前のファイルに設定をキャプチャします。

```
(New-EC2VpnConnection -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d).CustomerGatewayConfiguration | Out-File C:\path\vpn-
configuration.xml
```

例 3: この例では、指定された仮想プライベートゲートウェイと指定されたカスタマーゲートウェイの間に VPN 接続を作成します。

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d -Options_StaticRoutesOnly $true
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CreateVpnConnection](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定した仮想プライベートゲートウェイと指定したカスタマーゲートウェイの間に VPN 接続を作成します。出力には、ネットワーク管理者が必要とする設定情報が XML 形式で含まれます。

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d
```

出力:

```
CustomerGatewayConfiguration : [XML document]
CustomerGatewayId           : cgw-1a2b3c4d
Options                     :
```

```
Routes           : {}
State            : pending
Tags            : {}
Type            :
VgwTelemetry     : {}
VpnConnectionId  : vpn-12345678
VpnGatewayId     : vgw-1a2b3c4d
```

例 2: この例では、VPN 接続を作成し、指定された名前のファイルに設定をキャプチャします。

```
(New-EC2VpnConnection -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
vgw-1a2b3c4d).CustomerGatewayConfiguration | Out-File C:\path\vpn-
configuration.xml
```

例 3: この例では、指定された仮想プライベートゲートウェイと指定されたカスタマーゲートウェイの間に VPN 接続を作成します。

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
vgw-1a2b3c4d -Options_StaticRoutesOnly $true
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CreateVpnConnection](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **CreateVpnConnectionRoute** を使用する

次のサンプルコードは、CreateVpnConnectionRoute を使用方法を説明しています。

CLI

AWS CLI

VPN 接続の静的ルートを作成するには

この例では、指定された VPN 接続に対して静的ルートを作成しています。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 create-vpn-connection-route --vpn-connection-id vpn-40f41529 --  
destination-cidr-block 11.12.0.0/16
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateVpnConnectionRoute](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定された VPN 接続に対して指定された静的ルートを作成します。

```
New-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock  
11.12.0.0/16
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CreateVpnConnectionRoute](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定された VPN 接続に対して指定された静的ルートを作成します。

```
New-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock  
11.12.0.0/16
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CreateVpnConnectionRoute](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **CreateVpnGateway** を使用する

次のサンプルコードは、CreateVpnGateway を使用する方法を説明しています。

CLI

AWS CLI

仮想プライベートゲートウェイを作成するには

この例では、仮想プライベートゲートウェイを作成します。

コマンド:

```
aws ec2 create-vpn-gateway --type ipsec.1
```

出力:

```
{
  "VpnGateway": {
    "AmazonSideAsn": 64512,
    "State": "available",
    "Type": "ipsec.1",
    "VpnGatewayId": "vgw-9a4cacf3",
    "VpcAttachments": []
  }
}
```

特定の Amazon 側の ASN を使用して仮想プライベートゲートウェイを作成するには

この例では、仮想プライベートゲートウェイを作成し、BGP セッションの Amazon 側に AS 番号 (ASN) を指定します。

コマンド:

```
aws ec2 create-vpn-gateway --type ipsec.1 --amazon-side-asn 65001
```

出力:

```
{
  "VpnGateway": {
    "AmazonSideAsn": 65001,
    "State": "available",
    "Type": "ipsec.1",
    "VpnGatewayId": "vgw-9a4cacf3",
    "VpcAttachments": []
  }
}
```

```
}  
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[CreateVpnGateway](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定された仮想プライベートゲートウェイを作成します。

```
New-EC2VpnGateway -Type ipsec.1
```

出力:

```
AvailabilityZone :  
State           : available  
Tags            : {}  
Type            : ipsec.1  
VpcAttachments  : {}  
VpnGatewayId    : vgw-1a2b3c4d
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CreateVpnGateway](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定された仮想プライベートゲートウェイを作成します。

```
New-EC2VpnGateway -Type ipsec.1
```

出力:

```
AvailabilityZone :  
State           : available  
Tags            : {}  
Type            : ipsec.1  
VpcAttachments  : {}  
VpnGatewayId    : vgw-1a2b3c4d
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CreateVpnGateway](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DeleteCustomerGateway** を使用する

次のサンプルコードは、DeleteCustomerGateway を使用する方法を説明しています。

CLI

AWS CLI

カスタマーゲートウェイを削除するには

この例では、指定されたカスタマーゲートウェイを削除します。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 delete-customer-gateway --customer-gateway-id cgw-0e11f167
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteCustomerGateway](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたカスタマーゲートウェイを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

出力:

```
Confirm
```

```
Are you sure you want to perform this action?  
Performing operation "Remove-EC2CustomerGateway (DeleteCustomerGateway)" on  
Target "cgw-1a2b3c4d".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteCustomerGateway](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたカスタマーゲートウェイを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

出力:

```
Confirm  
Are you sure you want to perform this action?  
Performing operation "Remove-EC2CustomerGateway (DeleteCustomerGateway)" on  
Target "cgw-1a2b3c4d".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DeleteCustomerGateway](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DeleteDhcpOptions** を使用する

次のサンプルコードは、DeleteDhcpOptions を使用する方法を説明しています。

CLI

AWS CLI

DHCP オプションセットを削除するには

この例では、指定された DHCP オプションのセットを削除します。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 delete-dhcp-options --dhcp-options-id dopt-d9070ebb
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteDhcpOptions](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定された DHCP オプションのセットを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2DhcpOption (DeleteDhcpOptions)" on Target
"dopt-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteDhcpOptions](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定された DHCP オプションのセットを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d
```

出力:

```
Confirm
```



```
Are you sure you want to perform this action?  
Performing operation "Remove-EC2DhcpOption (DeleteDhcpOptions)" on Target  
"dopt-1a2b3c4d".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DeleteDhcpOptions](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DeleteFlowLogs** を使用する

次のサンプルコードは、DeleteFlowLogs を使用方法を説明しています。

CLI

AWS CLI

フローログを削除するには

次の delete-flow-logs の例では、指定したフローログを削除します。

```
aws ec2 delete-flow-logs --flow-log-id fl-11223344556677889
```

出力:

```
{  
  "Unsuccessful": []  
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteFlowLogs](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定された FlowLogId fl-01a2b3456a789c01 を削除します。

```
Remove-EC2FlowLog -FlowLogId fl-01a2b3456a789c01
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2FlowLog (DeleteFlowLogs)" on target
"fl-01a2b3456a789c01".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"): Y
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteFlowLogs](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定された FlowLogId fl-01a2b3456a789c01 を削除します。

```
Remove-EC2FlowLog -FlowLogId fl-01a2b3456a789c01
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2FlowLog (DeleteFlowLogs)" on target
"fl-01a2b3456a789c01".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"): Y
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DeleteFlowLogs](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DeleteInternetGateway** を使用する

次のサンプルコードは、DeleteInternetGateway を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [プライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成する](#)
- [Amazon VPC の使用を開始する](#)

CLI

AWS CLI

インターネットゲートウェイを削除するには

次の `delete-internet-gateway` の例では、指定したインターネットゲートウェイを削除します。

```
aws ec2 delete-internet-gateway \  
  --internet-gateway-id igw-0d0fb496b3EXAMPLE
```

このコマンドでは何も出力されません。

詳細については、Amazon VPC ユーザーガイドの「[インターネットゲートウェイ](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteInternetGateway](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたインターネットゲートウェイを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

出力:

```
Confirm  
Are you sure you want to perform this action?
```

```
Performing operation "Remove-EC2InternetGateway (DeleteInternetGateway)" on
Target "igw-1a2b3c4d".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteInternetGateway](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたインターネットゲートウェイを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2InternetGateway (DeleteInternetGateway)" on
Target "igw-1a2b3c4d".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DeleteInternetGateway](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DeleteKeyPair**で を使用する

次のサンプルコードは、DeleteKeyPair を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)
- [Amazon VPC の使用を開始する](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Delete an Amazon EC2 key pair.
/// </summary>
/// <param name="keyPairName">The name of the key pair to delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteKeyPair(string keyPairName)
{
    try
    {
        await _amazonEC2.DeleteKeyPairAsync(new
DeleteKeyPairRequest(keyPairName)).ConfigureAwait(false);
        return true;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidKeyPair.NotFound")
        {
            _logger.LogError($"KeyPair {keyPairName} does not exist and
cannot be deleted. Please verify the key pair name and try again.");
        }

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

/// <summary>
```

```

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

```

- APIの詳細については、AWS SDK for .NET API リファレンスの「[DeleteKeyPair](#)」を参照してください。

Bash

AWS CLI Bash スクリプトを使用する

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

#####
# 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
}

```

この例で使用されているユーティリティ関数。

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


- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteKeyPair](#)」を参照してください。

C++

SDK for C++

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
#!/ Delete an Amazon Elastic Compute Cloud (Amazon EC2) instance key pair.
/*!
    \param keyPairName: A name for a key pair.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/

bool AwsDoc::EC2::deleteKeyPair(const Aws::String &keyPairName,
                                const Aws::Client::ClientConfiguration
                                &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DeleteKeyPairRequest request;

    request.SetKeyName(keyPairName);
    const Aws::EC2::Model::DeleteKeyPairOutcome outcome =
ec2Client.DeleteKeyPair(
    request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to delete key pair " << keyPairName <<
            ":" << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully deleted key pair named " << keyPairName <<
            std::endl;
    }

    return outcome.IsSuccess();
}
```

- API の詳細については、AWS SDK for C++ API リファレンスの「[DeleteKeyPair](#)」を参照してください。

CLI

AWS CLI

キーペアを削除するには

次の delete-key-pair の例では、指定されたキーペアを削除します。

```
aws ec2 delete-key-pair \  
  --key-name my-key-pair
```

出力:

```
{  
  "Return": true,  
  "KeyId": "key-03c8d3aceb53b507"  
}
```

詳細については、「AWS コマンドラインインターフェイス」の「[Create and delete key pairs](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteKeyPair](#)」を参照してください。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/**
```

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

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

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[DeleteKeyPair](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

/**
 * Deletes the specified key pair, by removing the public key from Amazon EC2.
 * @param {{ keyName: string }} options
 */
export const main = async ({ keyName }) => {
  const client = new EC2Client({});
  const command = new DeleteKeyPairCommand({
    KeyName: keyName,
  });

  try {
    await client.send(command);
    console.log("Successfully deleted key pair.");
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide the required value?`);
    } else {
      throw caught;
    }
  }
};
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[DeleteKeyPair](#)」を参照してください。

Kotlin

SDK for Kotlin

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[DeleteKeyPair](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定したキーペアを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2KeyPair -KeyName my-key-pair
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2KeyPair (DeleteKeyPair)" on Target "my-key-pair".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteKeyPair](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定したキーペアを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2KeyPair -KeyName my-key-pair
```

出力:

Confirm

Are you sure you want to perform this action?

Performing operation "Remove-EC2KeyPair (DeleteKeyPair)" on Target "my-key-pair".

[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"):

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DeleteKeyPair](#)」を参照してください。

Python

SDK for Python (Boto3)

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
class KeyPairWrapper:
    """
    Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions.
    This class provides methods to create, list, and delete EC2 key pairs.
    """

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

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
        level
                           access to AWS EC2 services.
        :param key_file_dir: The folder where the private key information is
        stored.
```

```

        This should be a secure folder.
        :param key_pair: A dictionary representing the Boto3 KeyPair object.
                        This is a high-level object that wraps key pair actions.
Optional.
    """
    self.ec2_client = ec2_client
    self.key_pair = key_pair
    self.key_file_path: Optional[str] = None
    self.key_file_dir = key_file_dir

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

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

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

        :param key_name: The name of the key pair to delete.
        :return: A boolean indicating whether the deletion was successful.
        :raises ClientError: If there is an error in deleting the key pair, for
example,
                                if the key pair does not exist.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=key_name)
            logger.info(f"Successfully deleted key pair: {key_name}")
            self.key_pair = None
            return True
        except self.ec2_client.exceptions.ClientError as err:
            logger.error(f"Deletion failed for key pair: {key_name}")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidKeyPair.NotFound":
                logger.error(

```

```

        f"The key pair '{key_name}' does not exist and cannot be
deleted. "
        "Please verify the key pair name and try again."
    )
    raise

```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[DeleteKeyPair](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

バックアッププライベート PEM キーも削除する `delete_key` のラッパー。

```

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(())
}

```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[DeleteKeyPair](#)」を参照してください。

SAP ABAP

SDK for SAP ABAP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

TRY.
    lo_ec2->deletekeypair( iv_keyname = iv_key_name ).
    MESSAGE 'Amazon EC2 key pair deleted.' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.

```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「[DeleteKeyPair](#)」を参照してください。

Swift

SDK for Swift

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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
    }
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[DeleteKeyPair](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI `DeleteLaunchTemplate`で 使用する

次のサンプルコードは、`DeleteLaunchTemplate` を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [レジリエントなサービスの構築と管理](#)
- [プライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成する](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Delete a launch template by name.
/// </summary>
/// <param name="templateName">The name of the template to delete.</param>
/// <returns>Async task.</returns>
public async Task DeleteTemplateByName(string templateName)
{
    try
    {
        await _amazonEc2.DeleteLaunchTemplateAsync(
            new DeleteLaunchTemplateRequest()
            {
                LaunchTemplateName = templateName
            });
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode ==
            "InvalidLaunchTemplateName.NotFoundException")
        {
            _logger.LogError(
```

```
        $"Could not delete the template, the name
        {_launchTemplateName} was not found.");
    }

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

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[DeleteLaunchTemplate](#)」を参照してください。

CLI

AWS CLI

起動テンプレートを削除するには

次の例では、指定した起動テンプレートを削除しています。

コマンド:

```
aws ec2 delete-launch-template --launch-template-id lt-0abcd290751193123
```

出力:

```
{
  "LaunchTemplate": {
    "LatestVersionNumber": 2,
    "LaunchTemplateId": "lt-0abcd290751193123",
    "LaunchTemplateName": "TestTemplate",
    "DefaultVersionNumber": 2,
    "CreatedBy": "arn:aws:iam::123456789012:root",
    "CreateTime": "2017-11-23T16:46:25.000Z"
  }
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteLaunchTemplate](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
await client.send(  
    new DeleteLaunchTemplateCommand({  
        LaunchTemplateName: NAMES.launchTemplateName,  
    }),  
);
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[DeleteLaunchTemplate](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
class AutoScalingWrapper:  
    """  
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.  
    """  
  
    def __init__(
```

```

        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

```

```
def delete_template(self):
    """
    Deletes a launch template.
    """
    try:
        self.ec2_client.delete_launch_template(
            LaunchTemplateName=self.launch_template_name
        )
        self.delete_instance_profile(
            self.instance_profile_name, self.instance_role_name
        )
        log.info("Launch template %s deleted.", self.launch_template_name)
    except ClientError as err:
        if (
            err.response["Error"]["Code"]
            == "InvalidLaunchTemplateName.NotFoundException"
        ):
            log.info(
                "Launch template %s does not exist, nothing to do.",
                self.launch_template_name,
            )
        log.error(f"Full error:\n\t{err}")
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[DeleteLaunchTemplate](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DeleteNetworkAcl** を使用する

次のサンプルコードは、DeleteNetworkAcl を使用する方法を説明しています。

CLI

AWS CLI

ネットワーク ACL を削除するには

この例では、指定されたネットワーク ACL を削除します。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 delete-network-acl --network-acl-id acl-5fb85d36
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteNetworkAcl](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたネットワーク ACL を削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2NetworkAcl -NetworkAclId acl-12345678
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkAcl (DeleteNetworkAcl)" on Target
"acl-12345678".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteNetworkAcl](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたネットワーク ACL を削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。


```
Remove-EC2NetworkAcl -NetworkAclId acl-12345678
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkAcl (DeleteNetworkAcl)" on Target
"acl-12345678".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DeleteNetworkAcl](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DeleteNetworkAclEntry** を使用する

次のサンプルコードは、DeleteNetworkAclEntry を使用方法を説明しています。

CLI

AWS CLI

ネットワーク ACL エントリを削除するには

この例では、指定されたネットワーク ACL から受信ルール番号 100 を削除します。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 delete-network-acl-entry --network-acl-id acl-5fb85d36 --ingress --rule-  
number 100
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteNetworkAclEntry](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたネットワーク ACL から指定されたルールを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100
```

出力:

```
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"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteNetworkAclEntry](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたネットワーク ACL から指定されたルールを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100
```

出力:

```
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"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DeleteNetworkAclEntry](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で `DeleteNetworkInterface` を使用する

次のサンプルコードは、`DeleteNetworkInterface` を使用方法を説明しています。

CLI

AWS CLI

ネットワークインターフェイスを削除するには

この例では、指定されたネットワークインターフェイスを削除します。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 delete-network-interface --network-interface-id eni-e5aa89a3
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteNetworkInterface](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたネットワークインターフェイスを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkInterface (DeleteNetworkInterface)" on
Target "eni-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteNetworkInterface](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたネットワークインターフェイスを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkInterface (DeleteNetworkInterface)" on
Target "eni-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DeleteNetworkInterface](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DeletePlacementGroup** を使用する

次のサンプルコードは、DeletePlacementGroup を使用する方法を説明しています。

CLI

AWS CLI

プレイacementグループを削除するには

このコマンド例では、指定されたプレイacementグループを削除します。

コマンド:

```
aws ec2 delete-placement-group --group-name my-cluster
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeletePlacementGroup](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたプレイacementグループを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2PlacementGroup -GroupName my-placement-group
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2PlacementGroup (DeletePlacementGroup)" on Target
"my-placement-group".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeletePlacementGroup](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたプレイacementグループを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2PlacementGroup -GroupName my-placement-group
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2PlacementGroup (DeletePlacementGroup)" on Target
"my-placement-group".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DeletePlacementGroup](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DeleteRoute** を使用する

次のサンプルコードは、DeleteRoute を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [Transit Gateway を開始する](#)

CLI

AWS CLI

ルートを削除するには

この例では、指定したルートテーブルから指定したルートを削除します。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 delete-route --route-table-id rtb-22574640 --destination-cidr-block 0.0.0.0/0
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteRoute](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定したルートテーブルから指定したルートを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Route (DeleteRoute)" on Target "rtb-1a2b3c4d".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteRoute](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定したルートテーブルから指定したルートを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Route (DeleteRoute)" on Target "rtb-1a2b3c4d".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DeleteRoute](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DeleteRouteTable** を使用する

次のサンプルコードは、DeleteRouteTable を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [プライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成する](#)
- [Amazon VPC の使用を開始する](#)

CLI

AWS CLI

ルートテーブルを削除するには

この例では、指定されたルートテーブルを削除します。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 delete-route-table --route-table-id rtb-22574640
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteRouteTable](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたルートテーブルを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2RouteTable (DeleteRouteTable)" on Target
"rtb-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteRouteTable](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたルートテーブルを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2RouteTable (DeleteRouteTable)" on Target
"rtb-1a2b3c4d".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DeleteRouteTable](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DeleteSecurityGroup**で 使用する

次のサンプルコードは、DeleteSecurityGroup を使用方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)
- [プライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成する](#)
- [Amazon VPC の使用を開始する](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

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

- API の詳細については、AWS SDK for .NET API リファレンスの「[DeleteSecurityGroup](#)」を参照してください。

Bash

AWS CLI Bash スクリプトを使用する

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
#####
# 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
}

```

この例で使用されているユーティリティ関数。

```

#####
# 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
}
```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[DeleteSecurityGroup](#)」を参照してください。

C++

SDK for C++

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
//! Delete a security group.
/*!
    \param securityGroupID: A security group ID.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
 */
bool AwsDoc::EC2::deleteSecurityGroup(const Aws::String &securityGroupID,
                                       const Aws::Client::ClientConfiguration
                                       &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DeleteSecurityGroupRequest request;

    request.SetGroupId(securityGroupID);
    Aws::EC2::Model::DeleteSecurityGroupOutcome outcome =
    ec2Client.DeleteSecurityGroup(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to delete security group " << securityGroupID <<
            ":" << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully deleted security group " << securityGroupID <<
            std::endl;
    }

    return outcome.IsSuccess();
}
```

- API の詳細については、AWS SDK for C++ API リファレンスの「[DeleteSecurityGroup](#)」を参照してください。

CLI

AWS CLI

[EC2-Classic] セキュリティグループを削除するには

この例では、MySecurityGroup という名前のセキュリティグループを削除します。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 delete-security-group --group-name MySecurityGroup
```

[EC2-VPC] セキュリティグループを削除するには

この例では、sg-903004f8 という ID のセキュリティグループを削除します。EC2-VPC 用セキュリティグループは名前では参照できないことに注意してください。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 delete-security-group --group-id sg-903004f8
```

詳細については、「AWS コマンドラインインターフェイスユーザーガイド」でセキュリティグループの使用方法を参照してください。

- API の詳細については、「AWS CLI Command Reference」の「[DeleteSecurityGroup](#)」を参照してください。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/**  
 * Deletes an EC2 security group asynchronously.
```

```

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

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

```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[DeleteSecurityGroup](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

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

/**
 * Deletes a security group.

```



```
* @param {{ groupId: string }} options
*/
export const main = async ({ groupId }) => {
  const client = new EC2Client({});
  const command = new DeleteSecurityGroupCommand({
    GroupId: groupId,
  });

  try {
    await client.send(command);
    console.log("Security group deleted successfully.");
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidGroupId.Malformed") {
      console.warn(`${caught.message}. Please provide a valid GroupId.`);
    } else {
      throw caught;
    }
  }
};
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[DeleteSecurityGroup](#)」を参照してください。

Kotlin

SDK for Kotlin

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[DeleteSecurityGroup](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、EC2-VPC の指定されたセキュリティグループを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2SecurityGroup -GroupId sg-12345678
```

出力:

```
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"):
```

例 2: この例では、EC2-Classic の指定されたセキュリティグループを削除します。

```
Remove-EC2SecurityGroup -GroupName my-security-group -Force
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteSecurityGroup](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、EC2-VPC の指定されたセキュリティグループを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2SecurityGroup -GroupId sg-12345678
```

出力:

```
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"):
```

例 2: この例では、EC2-Classic の指定されたセキュリティグループを削除します。

```
Remove-EC2SecurityGroup -GroupName my-security-group -Force
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DeleteSecurityGroup](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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

```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[DeleteSecurityGroup](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

pub async fn delete_security_group(&self, group_id: &str) -> Result<(),
EC2Error> {
    tracing::info!("Deleting security group {group_id}");
    self.client
        .delete_security_group()
        .group_id(group_id)
        .send()
        .await?;
    Ok(())
}

```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[DeleteSecurityGroup](#)」を参照してください。

SAP ABAP

SDK for SAP ABAP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
TRY.  
    lo_ec2->deletesecuritygroup( iv_groupid = iv_security_group_id ).  
    MESSAGE 'Security group deleted.' TYPE 'I'.  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception->av_err_msg }|.  
    MESSAGE lv_error TYPE 'E'.  
ENDTRY.
```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「[DeleteSecurityGroup](#)」を参照してください。

Swift

SDK for Swift

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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
    }
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[DeleteSecurityGroup](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DeleteSnapshot**で 使用する

次のサンプルコードは、DeleteSnapshot を使用方法を説明しています。

CLI

AWS CLI

スナップショットを削除するには

このコマンド例は、スナップショット ID が snap-1234567890abcdef0 のスナップショットを削除します。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 delete-snapshot --snapshot-id snap-1234567890abcdef0
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteSnapshot](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: こので例は、指定したスナップショットを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2Snapshot -SnapshotId snap-12345678
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Snapshot (DeleteSnapshot)" on target
"snap-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteSnapshot](#)」を参照してください。

Tools for PowerShell V5

例 1: こので例は、指定したスナップショットを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2Snapshot -SnapshotId snap-12345678
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Snapshot (DeleteSnapshot)" on target
"snap-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):
```


- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DeleteSnapshot](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
async fn delete_snapshot(client: &Client, id: &str) -> Result<(), Error> {
    client.delete_snapshot().snapshot_id(id).send().await?;

    println!("Deleted");

    Ok(())
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[DeleteSnapshot](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DeleteSpotDatafeedSubscription** を使用する

次のサンプルコードは、DeleteSpotDatafeedSubscription を使用方法を説明しています。

CLI

AWS CLI

スポットインスタンスデータフィードサブスクリプションをキャンセルするには

このコマンド例では、アカウントのスポットデータフィードサブスクリプションを削除します。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 delete-spot-datafeed-subscription
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteSpotDatafeedSubscription](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、スポットインスタンスのデータフィードを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2SpotDatafeedSubscription
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2SpotDatafeedSubscription
(DeleteSpotDatafeedSubscription)" on Target "".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteSpotDatafeedSubscription](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、スポットインスタンスのデータフィードを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2SpotDatafeedSubscription
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2SpotDatafeedSubscription
(DeleteSpotDatafeedSubscription)" on Target "".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DeleteSpotDatafeedSubscription](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DeleteSubnet** を使用する

次のサンプルコードは、DeleteSubnet を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [プライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成する](#)
- [Amazon VPC の使用を開始する](#)

CLI

AWS CLI

サブネットを削除するには

この例では、指定されたサブネットを削除します。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 delete-subnet --subnet-id subnet-9d4a7b6c
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteSubnet](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定したサブネットを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2Subnet -SubnetId subnet-1a2b3c4d
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Subnet (DeleteSubnet)" on Target
"subnet-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteSubnet](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定したサブネットを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2Subnet -SubnetId subnet-1a2b3c4d
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Subnet (DeleteSubnet)" on Target
"subnet-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DeleteSubnet](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で DeleteTags を使用する

次のサンプルコードは、DeleteTags を使用方法を説明しています。

CLI

AWS CLI

例 1: リソースからタグを削除するには

次の delete-tags の例では、指定したイメージからタグ Stack=Test を削除します。値とキー名の両方を指定すると、タグの値が指定された値と一致する場合にのみ、タグは削除されます。

```
aws ec2 delete-tags \  
  --resources ami-1234567890abcdef0 \  
  --tags Key=Stack,Value=Test
```

タグの値を指定することはオプションです。次の delete-tags の例では、タグのタグ値に関係なく、指定したインスタンスからキー名 purpose を持つタグを削除します。

```
aws ec2 delete-tags \  
  --resources i-1234567890abcdef0 \  
  --tags Key=purpose
```

空の文字列をタグ値として指定すると、タグの値が空の文字列である場合にのみタグが削除されます。次の delete-tags の例では、空の文字列を、削除するタグのタグ値として指定します。

```
aws ec2 delete-tags \  
  --resources i-1234567890abcdef0 \  
  --tags Key=Name,Value=
```

例 2: 複数のリソースからタグを削除するには

次の delete-tags の例では、インスタンスと AMI の両方からタグ「Purpose=Test」を削除します。前の例に示されているように、コマンドのタグ値を省略できます。

```
aws ec2 delete-tags \  
  --resources i-1234567890abcdef0 ami-1234567890abcdef0 \  
  --tags Key=Purpose
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteTags](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、タグ値に関係なく、指定されたリソースから指定されたタグを削除します。この例で使用される構文には、PowerShell バージョン 3 以降が必要です。

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag" } -Force
```

例 2: この例では、指定されたタグ値が一致する場合にのみ、指定されたリソースから指定されたタグを削除します。この例で使用される構文には、PowerShell バージョン 3 以降が必要です。

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag";Value="myTagValue" } -Force
```

例 3: この例では、タグ値に関係なく、指定されたリソースから指定されたタグを削除します。

```
$tag = New-Object Amazon.EC2.Model.Tag  
$tag.Key = "myTag"  
  
Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

例 4: この例では、指定されたタグ値が一致する場合にのみ、指定されたリソースから指定されたタグを削除します。

```
$tag = New-Object Amazon.EC2.Model.Tag  
$tag.Key = "myTag"  
$tag.Value = "myTagValue"  
  
Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteTags](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、タグ値に関係なく、指定されたリソースから指定されたタグを削除します。この例で使用される構文には、PowerShell バージョン 3 以降が必要です。

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag" } -Force
```

例 2: この例では、指定されたタグ値が一致する場合にのみ、指定されたリソースから指定されたタグを削除します。この例で使用される構文には、PowerShell バージョン 3 以降が必要です。

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag";Value="myTagValue" } -Force
```

例 3: この例では、タグ値に関係なく、指定されたリソースから指定されたタグを削除します。

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

例 4: この例では、指定されたタグ値が一致する場合にのみ、指定されたリソースから指定されたタグを削除します。

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DeleteTags](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DeleteVolume** を使用する

次のサンプルコードは、DeleteVolume を使用する方法を説明しています。

CLI

AWS CLI

ボリュームを削除するには

このコマンド例では、vol-049df61146c4d7901 というボリューム ID を持つ使用可能なボリュームを削除します。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 delete-volume --volume-id vol-049df61146c4d7901
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteVolume](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたボリュームをデタッチします。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2Volume -VolumeId vol-12345678
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Volume (DeleteVolume)" on target
"vol-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteVolume](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたボリュームをデタッチします。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2Volume -VolumeId vol-12345678
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Volume (DeleteVolume)" on target
"vol-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DeleteVolume](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DeleteVpc**で 使用する

次のサンプルコードは、DeleteVpc を使用方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [プライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成する](#)
- [Amazon VPC の使用を開始する](#)
- [VPC IPAM の開始方法](#)

CLI

AWS CLI

VPC を削除するには

この例では、指定された VPC を削除します。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 delete-vpc --vpc-id vpc-a01106c2
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteVpc](#)」を参照してください。

PHP

SDK for PHP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/**
 * @param string $vpcId
 * @return void
 */
public function deleteVpc(string $vpcId)
{
    try {
        $this->ec2Client->deleteVpc([
            "VpcId" => $vpcId,
        ]);
    } catch (Ec2Exception $caught) {
        echo "There was a problem deleting the VPC: {$caught->getAwsErrorMessage()}\n";
        throw $caught;
    }
}
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「[DeleteVpc](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定した VPC を削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2Vpc -VpcId vpc-12345678
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Vpc (DeleteVpc)" on Target "vpc-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteVpc](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定した VPC を削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2Vpc -VpcId vpc-12345678
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Vpc (DeleteVpc)" on Target "vpc-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DeleteVpc](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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

```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[DeleteVpc](#)」を参照してください。

SAP ABAP

SDK for SAP ABAP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

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.

```

- API の詳細については、AWS SDK for SAP ABAP API リファレンスの[DeleteVpc](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DeleteVpcEndpoints**で を使用する

次のサンプルコードは、DeleteVpcEndpoints を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [プライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成する](#)

CLI

AWS CLI

エンドポイントを削除するには

この例では、エンドポイント vpce-aa22bb33 と vpce-1a2b3c4d を削除します。コマンドが部分的に成功または失敗した場合は、失敗した項目のリストが返されます。コマンドが成功した場合、返されたリストは空になります。

コマンド:

```
aws ec2 delete-vpc-endpoints --vpc-endpoint-ids vpce-aa22bb33 vpce-1a2b3c4d
```

出力:

```
{
  "Unsuccessful": []
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteVpcEndpoints](#)」を参照してください。

PHP

SDK for PHP

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/**
 * @param string $vpcEndpointId
 * @return void
 */
public function deleteVpcEndpoint(string $vpcEndpointId)
{
    try {
        $this->ec2Client->deleteVpcEndpoints([
            "VpcEndpointIds" => [$vpcEndpointId],
        ]);
    } catch (Ec2Exception $caught){
        echo "There was a problem deleting the VPC Endpoint: {$caught->getAwsErrorMessage()}\n";
        throw $caught;
    }
}
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「[DeleteVpcEndpoints](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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

```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[DeleteVpcEndpoints](#)」を参照してください。

SAP ABAP

SDK for SAP ABAP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

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.

```

- API の詳細については、AWS SDK for SAP ABAP API リファレンスの[DeleteVpcEndpoints](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で `DeleteVpnConnection` を使用する

次のサンプルコードは、`DeleteVpnConnection` を使用方法を説明しています。

CLI

AWS CLI

VPN 接続を削除するには

この例では、指定された VPN 接続を削除します。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 delete-vpn-connection --vpn-connection-id vpn-40f41529
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteVpnConnection](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定された VPN 接続を削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2VpnConnection -VpnConnectionId vpn-12345678
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2VpnConnection (DeleteVpnConnection)" on Target
"vpn-12345678".
```

```
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteVpnConnection](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定された VPN 接続を削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2VpnConnection -VpnConnectionId vpn-12345678
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2VpnConnection (DeleteVpnConnection)" on Target
"vpn-12345678".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DeleteVpnConnection](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DeleteVpnConnectionRoute** を使用する

次のサンプルコードは、DeleteVpnConnectionRoute を使用方法を説明しています。

CLI

AWS CLI

VPN 接続から静的ルートを削除するには

この例では、指定された VPN 接続から指定された静的ルートを削除します。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 delete-vpn-connection-route --vpn-connection-id vpn-40f41529 --  
destination-cidr-block 11.12.0.0/16
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteVpnConnectionRoute](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定された VPN 接続から指定された静的ルートを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock  
11.12.0.0/16
```

出力:

```
Confirm  
Are you sure you want to perform this action?  
Performing operation "Remove-EC2VpnConnectionRoute (DeleteVpnConnectionRoute)" on  
Target "vpn-12345678".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteVpnConnectionRoute](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定された VPN 接続から指定された静的ルートを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock  
11.12.0.0/16
```

出力:

Confirm

Are you sure you want to perform this action?

Performing operation "Remove-EC2VpnConnectionRoute (DeleteVpnConnectionRoute)" on Target "vpn-12345678".

[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"):

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DeleteVpnConnectionRoute](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DeleteVpnGateway** を使用する

次のサンプルコードは、DeleteVpnGateway を使用する方法を説明しています。

CLI

AWS CLI

仮想プライベートゲートウェイを削除するには

この例では、指定された仮想プライベートゲートウェイを削除します。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 delete-vpn-gateway --vpn-gateway-id vgw-9a4cacf3
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeleteVpnGateway](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定された仮想プライベートゲートウェイを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2VpnGateway (DeleteVpnGateway)" on Target
"vgw-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteVpnGateway](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定された仮想プライベートゲートウェイを削除します。Force パラメータも指定しない限り、操作を続行する前に確認を求めるプロンプトが表示されます。

```
Remove-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2VpnGateway (DeleteVpnGateway)" on Target
"vgw-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DeleteVpnGateway](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DeregisterImage** を使用する

次のサンプルコードは、DeregisterImage を使用する方法を説明しています。

CLI

AWS CLI

AMI の登録を解除するには

この例では、指定された AMI の登録を解除します。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 deregister-image --image-id ami-4fa54026
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DeregisterImage](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定された AMI の登録を解除します。

```
Unregister-EC2Image -ImageId ami-12345678
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeregisterImage](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定された AMI の登録を解除します。

```
Unregister-EC2Image -ImageId ami-12345678
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DeregisterImage](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeAccountAttributes** を使用する

次のサンプルコードは、DescribeAccountAttributes を使用する方法を説明しています。

CLI

AWS CLI

AWS アカウントのすべての属性を記述するには

この例では、AWS アカウントの属性について説明します。

コマンド:

```
aws ec2 describe-account-attributes
```

出力:

```
{
  "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"
      }
    ]
  }
]
}

```

AWS アカウントの単一の属性を記述するには

この例では、AWS アカウントの `supported-platforms` 属性について説明します。

コマンド:

```
aws ec2 describe-account-attributes --attribute-names supported-platforms
```

出力:

```

{
  "AccountAttributes": [

```

```
{
  "AttributeName": "supported-platforms",
  "AttributeValues": [
    {
      "AttributeValue": "EC2"
    },
    {
      "AttributeValue": "VPC"
    }
  ]
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeAccountAttributes](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、リージョンの EC2-Classic および EC2-VPC にインスタンスを起動できるか、または EC2-VPC にのみインスタンスを起動できるかを記述します。

```
(Get-EC2AccountAttribute -AttributeName supported-platforms).AttributeValues
```

出力:

```
AttributeValue
-----
EC2
VPC
```

例 2: この例では、デフォルトの VPC を記述します。リージョンにデフォルトの VPC がない場合は「none」です。

```
(Get-EC2AccountAttribute -AttributeName default-vpc).AttributeValues
```

出力:

```
AttributeValue
```

```
-----  
vpc-12345678
```

例 3: この例では、実行できるオンデマンドインスタンスの最大数を記述します。

```
(Get-EC2AccountAttribute -AttributeName max-instances).AttributeValues
```

出力:

```
AttributeValue  
-----  
20
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeAccountAttributes](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、リージョンの EC2-Classic および EC2-VPC にインスタンスを起動できるか、または EC2-VPC にのみインスタンスを起動できるかを記述します。

```
(Get-EC2AccountAttribute -AttributeName supported-platforms).AttributeValues
```

出力:

```
AttributeValue  
-----  
EC2  
VPC
```

例 2: この例では、デフォルトの VPC を記述します。リージョンにデフォルトの VPC がない場合は「none」です。

```
(Get-EC2AccountAttribute -AttributeName default-vpc).AttributeValues
```

出力:

```
AttributeValue  
-----  
vpc-12345678
```

例 3: この例では、実行できるオンデマンドインスタンスの最大数を記述します。

```
(Get-EC2AccountAttribute -AttributeName max-instances).AttributeValues
```

出力:

```
AttributeValue
-----
20
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeAccountAttributes](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DescribeAddresses**で 使用する

次のサンプルコードは、DescribeAddresses を使用方法を説明しています。

C++

SDK for C++

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
#!/ Describe all Elastic IP addresses.
/*!
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::EC2::describeAddresses(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeAddressesRequest request;
```

```

    Aws::EC2::Model::DescribeAddressesOutcome outcome =
ec2Client.DescribeAddresses(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left << std::setw(20) << "InstanceId" <<
            std::setw(15) << "Public IP" << std::setw(10) << "Domain" <<
            std::setw(30) << "Allocation ID" << std::setw(25) <<
            "NIC ID" << std::endl;

        const Aws::Vector<Aws::EC2::Model::Address> &addresses =
outcome.GetResult().GetAddresses();
        for (const auto &address: addresses) {
            Aws::String domainString =
                Aws::EC2::Model::DomainTypeMapper::GetNameForDomainType(
                    address.GetDomain());

            std::cout << std::left << std::setw(20) <<
                address.GetInstanceId() << std::setw(15) <<
                address.GetPublicIp() << std::setw(10) << domainString <<
                std::setw(30) << address.GetAllocationId() << std::setw(25)
                << address.GetNetworkInterfaceId() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe Elastic IP addresses:" <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- APIの詳細については、「AWS SDK for C++ API リファレンス」の「[DescribeAddresses](#)」を参照してください。

CLI

AWS CLI

例 1: すべての Elastic IP アドレスに関する詳細を取得するには

次の describe addresses の例では、Elastic IP アドレスに関する詳細が表示されます。

```
aws ec2 describe-addresses
```

出力:

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

例 2: EC2-VPC の Elastic IP アドレスに関する詳細を取得するには

次の describe-addresses の例では、VPC 内のインスタンスで使用する Elastic IP アドレスの詳細が表示されます。

```
aws ec2 describe-addresses \
  --filters "Name=domain,Values=vpc"
```

出力:

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

例 3: 割り当て ID で指定された Elastic IP アドレスに関する詳細を取得するには

次の describe-addresses の例では、EC2-VPC 内のインスタンスに関連付けられている、指定された割り当て ID を持つ Elastic IP アドレスの詳細を表示します。

```
aws ec2 describe-addresses \
    --allocation-ids eipalloc-282d9641
```

出力:

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

例 4: VPC プライベート IP アドレスで指定された Elastic IP アドレスに関する詳細を取得するには

次の describe-addresses の例では、EC2-VPC の特定のプライベート IP アドレスに関連付けられている Elastic IP アドレスの詳細が表示されます。

```
aws ec2 describe-addresses \
```

```
--filters "Name=private-ip-address,Values=10.251.50.12"
```

例 5: EC2-Classic の Elastic IP アドレスに関する詳細を取得するには

次の describe-addresses の例では、EC2-Classic で使用する Elastic IP アドレスを表示します。

```
aws ec2 describe-addresses \  
  --filters "Name=domain,Values=standard"
```

出力:

```
{  
  "Addresses": [  
    {  
      "InstanceId": "i-1234567890abcdef0",  
      "PublicIp": "203.0.110.25",  
      "PublicIpv4Pool": "amazon",  
      "Domain": "standard"  
    }  
  ]  
}
```

例 6: パブリック IP アドレスで指定された Elastic IP アドレスに関する詳細を取得するには

次の describe-addresses の例では、EC2-Classic のインスタンスに関連付けられている、値 203.0.110.25 を持つ Elastic IP アドレスの詳細を表示します。

```
aws ec2 describe-addresses \  
  --public-ips 203.0.110.25
```

出力:

```
{  
  "Addresses": [  
    {  
      "InstanceId": "i-1234567890abcdef0",  
      "PublicIp": "203.0.110.25",  
      "PublicIpv4Pool": "amazon",  
      "Domain": "standard"  
    }  
  ]  
}
```



```
]
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeAddresses](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

/**
 * Describes the specified Elastic IP addresses or all of your Elastic IP
 * addresses.
 * @param {{ allocationId: string }} options
 */
export const main = async ({ allocationId }) => {
  const client = new EC2Client({});
  const command = new DescribeAddressesCommand({
    // You can omit this property to show all addresses.
    AllocationIds: [allocationId],
  });

  try {
    const { Addresses } = await client.send(command);
    const addressList = Addresses.map((address) => ` • ${address.PublicIp}`);
    console.log("Elastic IP addresses:");
    console.log(addressList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidAllocationID.NotFound"
    ) {
      console.warn(`${caught.message}. Please provide a valid AllocationId.`);
    }
  }
}
```

```
    } else {  
        throw caught;  
    }  
}  
};
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[DescribeAddresses](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、EC2-Classic のインスタンスに指定された Elastic IP アドレスを記述します。

```
Get-EC2Address -AllocationId eipalloc-12345678
```

出力:

```
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
```

例 2: この例では、VPC 内のインスタンスの Elastic IP アドレスを記述します。この構文には、PowerShell バージョン 3 以降が必要です。

```
Get-EC2Address -Filter @{ Name="domain";Values="vpc" }
```

例 3: この例では、EC2-Classic のインスタンスに指定された Elastic IP アドレスを記述します。

```
Get-EC2Address -PublicIp 203.0.113.17
```

出力:

```
AllocationId      :
AssociationId     :
Domain           : standard
InstanceId        : i-12345678
NetworkInterfaceId :
NetworkInterfaceOwnerId :
PrivateIpAddress  :
PublicIp         : 203.0.113.17
```

例 4: この例では、EC2-Classic のインスタンスの Elastic IP アドレスを記述します。この構文には、PowerShell バージョン 3 以降が必要です。

```
Get-EC2Address -Filter @{ Name="domain";Values="standard" }
```

例 5: この例では、すべての Elastic IP アドレスを記述します。

```
Get-EC2Address
```

例 6: この例では、フィルターで指定されたインスタンス ID のパブリック IP とプライベート IP を返します。

```
Get-EC2Address -Region eu-west-1 -Filter @{Name="instance-id";Values="i-0c12d3f4f567ffb89"} | Select-Object PrivateIpAddress, PublicIp
```

出力:

```
PrivateIpAddress PublicIp
-----
10.0.0.99         63.36.5.227
```

例 7: この例では、割り当て ID、関連付け ID、インスタンス ID を持つすべての Elastic IP を取得します。

```
Get-EC2Address -Region eu-west-1 | Select-Object InstanceId, AssociationId, AllocationId, PublicIp
```

出力:

InstanceId PublicIp ----- -----	AssociationId -----	AllocationId -----
		eipalloc-012e3b456789e1fad
17.212.120.178 i-0c123dfd3415bac67	eipassoc-0e123456bb7890bdb	eipalloc-01cd23ebf45f7890c
17.212.124.77		eipalloc-012345678eeabcfad
17.212.225.7 i-0123d405c67e89a0c	eipassoc-0c123b456783966ba	eipalloc-0123cdd456a8f7892
37.216.52.173 i-0f1bf2f34c5678d09	eipassoc-0e12934568a952d96	eipalloc-0e1c23e4d5e6789e4
37.218.222.278 i-012e3cb4df567e8aa	eipassoc-0d1b2fa4d67d03810	eipalloc-0123f456f78a01b58
37.210.82.27 i-0123bcf4b567890e1	eipassoc-01d2345f678903fb1	eipalloc-0e1db23cfef5c45c7
37.215.222.270		

例 8: この例では、タグキー「Category」と値「Prod」に一致する EC2 IP アドレスのリストを取得します。

```
Get-EC2Address -Filter @{"Name"="tag:Category";Values="Prod"}
```

出力:

```
AllocationId      : eipalloc-0123f456f81a01b58
AssociationId     : eipassoc-0d1b23a456d103810
CustomerOwnedIp   :
CustomerOwnedIpv4Pool :
Domain           : vpc
InstanceId        : i-012e3cb4df567e1aa
NetworkBorderGroup : eu-west-1
NetworkInterfaceId : eni-0123f41d5a60d5f40
NetworkInterfaceOwnerId : 123456789012
PrivateIpAddress  : 192.168.1.84
PublicIp          : 34.250.81.29
PublicIpv4Pool    : amazon
Tags              : {Category, Name}
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeAddresses](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、EC2-Classic のインスタンスに指定された Elastic IP アドレスを記述します。

```
Get-EC2Address -AllocationId eipalloc-12345678
```

出力:

```
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
```

例 2: この例では、VPC 内のインスタンスの Elastic IP アドレスを記述します。この構文には、PowerShell バージョン 3 以降が必要です。

```
Get-EC2Address -Filter @{ Name="domain";Values="vpc" }
```

例 3: この例では、EC2-Classic のインスタンスに指定された Elastic IP アドレスを記述します。

```
Get-EC2Address -PublicIp 203.0.113.17
```

出力:

```
AllocationId      :
AssociationId     :
Domain           : standard
InstanceId        : i-12345678
NetworkInterfaceId :
NetworkInterfaceOwnerId :
PrivateIpAddress  :
PublicIp         : 203.0.113.17
```

例 4: この例では、EC2-Classic のインスタンスの Elastic IP アドレスを記述します。この構文には、PowerShell バージョン 3 以降が必要です。

```
Get-EC2Address -Filter @{ Name="domain";Values="standard" }
```

例 5: この例では、すべての Elastic IP アドレスを記述します。

```
Get-EC2Address
```

例 6: この例では、フィルターで指定されたインスタンス ID のパブリック IP とプライベート IP を返します。

```
Get-EC2Address -Region eu-west-1 -Filter @{Name="instance-  
id";Values="i-0c12d3f4f567ffb89"} | Select-Object PrivateIpAddress, PublicIp
```

出力:

```
PrivateIpAddress PublicIp  
-----  
10.0.0.99          63.36.5.227
```

例 7: この例では、割り当て ID、関連付け ID、インスタンス ID を持つすべての Elastic IP を取得します。

```
Get-EC2Address -Region eu-west-1 | Select-Object InstanceId, AssociationId,  
AllocationId, PublicIp
```

出力:

```
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
```

例 8: この例では、タグキー「Category」と値「Prod」に一致する EC2 IP アドレスのリストを取得します。

```
Get-EC2Address -Filter @{"Name"="tag:Category";Values="Prod"}
```

出力:

```
AllocationId      : eipalloc-0123f456f81a01b58
AssociationId     : eipassoc-0d1b23a456d103810
CustomerOwnedIp   :
CustomerOwnedIpv4Pool :
Domain           : vpc
InstanceId        : i-012e3cb4df567e1aa
NetworkBorderGroup : eu-west-1
NetworkInterfaceId : eni-0123f41d5a60d5f40
NetworkInterfaceOwnerId : 123456789012
PrivateIpAddress  : 192.168.1.84
PublicIp          : 34.250.81.29
PublicIpv4Pool    : amazon
Tags              : {Category, Name}
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeAddresses](#)」を参照してください。

SAP ABAP

SDK for SAP ABAP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

TRY.

```

        oo_result = lo_ec2->describeaddresses( ).
        oo_result is returned for testing purposes. "
        DATA(lt_addresses) = oo_result->get_addresses( ).
        MESSAGE 'Retrieved information about Elastic IP addresses.' TYPE 'I'.
        CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.

```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「[DescribeAddresses](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DescribeAvailabilityZones**で を使用する

次のサンプルコードは、DescribeAvailabilityZones を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [レジリエントなサービスの構築と管理](#)
- [プライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成する](#)
- [Amazon VPC の使用を開始する](#)
- [Transit Gateway を開始する](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。


```
/// <summary>
/// Get a list of Availability Zones in the AWS Region of the Amazon EC2
Client.
/// </summary>
/// <returns>A list of availability zones.</returns>
public async Task<List<string>> DescribeAvailabilityZones()
{
    try
    {
        var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
            new DescribeAvailabilityZonesRequest());
        return zoneResponse.AvailabilityZones.Select(z =>
z.ZoneName).ToList();
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        _logger.LogError($"An Amazon EC2 error occurred while listing
availability zones.: {ec2Exception.Message}");
        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError($"An error occurred while listing availability
zones.: {ex.Message}");
        throw;
    }
}
```

- APIの詳細については、AWS SDK for .NET API リファレンスの「[DescribeAvailabilityZones](#)」を参照してください。

C++

SDK for C++

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

//! DescribeAvailabilityZones
/*!
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
int AwsDoc::EC2::describeAvailabilityZones(const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeAvailabilityZonesRequest request;
    Aws::EC2::Model::DescribeAvailabilityZonesOutcome outcome =
ec2Client.DescribeAvailabilityZones(request);

    if (outcome.IsSuccess()) {
        std::cout << std::left <<
            std::setw(32) << "ZoneName" <<
            std::setw(20) << "State" <<
            std::setw(32) << "Region" << std::endl;

        const auto &zones =
            outcome.GetResult().GetAvailabilityZones();

        for (const auto &zone: zones) {
            Aws::String stateString =

Aws::EC2::Model::AvailabilityZoneStateMapper::GetNameForAvailabilityZoneState(
                zone.GetState());
            std::cout << std::left <<
                std::setw(32) << zone.GetZoneName() <<
                std::setw(20) << stateString <<
                std::setw(32) << zone.GetRegionName() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe availability zones:" <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API の詳細については、AWS SDK for C++ API リファレンスの「[DescribeAvailabilityZones](#)」を参照してください。

CLI

AWS CLI

アベイラビリティゾーンを説明するには

次の describe-availability-zones の例では、利用可能なアベイラビリティゾーンの詳細が表示されます。レスポンスには、現在のリージョンのアベイラビリティゾーンのみが含まれます。この例では、デフォルトの us-west-2 (オレゴン) リージョンのプロファイルを使用しています。

```
aws ec2 describe-availability-zones
```

出力:

```
{
  "AvailabilityZones": [
    {
      "State": "available",
      "OptInStatus": "opt-in-not-required",
      "Messages": [],
      "RegionName": "us-west-2",
      "ZoneName": "us-west-2a",
      "ZoneId": "usw2-az1",
      "GroupName": "us-west-2",
      "NetworkBorderGroup": "us-west-2"
    },
    {
      "State": "available",
      "OptInStatus": "opt-in-not-required",
      "Messages": [],
      "RegionName": "us-west-2",
      "ZoneName": "us-west-2b",
      "ZoneId": "usw2-az2",
      "GroupName": "us-west-2",
      "NetworkBorderGroup": "us-west-2"
    },
    {
      "State": "available",
```

```

        "OptInStatus": "opt-in-not-required",
        "Messages": [],
        "RegionName": "us-west-2",
        "ZoneName": "us-west-2c",
        "ZoneId": "usw2-az3",
        "GroupName": "us-west-2",
        "NetworkBorderGroup": "us-west-2"
    },
    {
        "State": "available",
        "OptInStatus": "opt-in-not-required",
        "Messages": [],
        "RegionName": "us-west-2",
        "ZoneName": "us-west-2d",
        "ZoneId": "usw2-az4",
        "GroupName": "us-west-2",
        "NetworkBorderGroup": "us-west-2"
    },
    {
        "State": "available",
        "OptInStatus": "opted-in",
        "Messages": [],
        "RegionName": "us-west-2",
        "ZoneName": "us-west-2-lax-1a",
        "ZoneId": "usw2-lax1-az1",
        "GroupName": "us-west-2-lax-1",
        "NetworkBorderGroup": "us-west-2-lax-1"
    }
]
}

```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[DescribeAvailabilityZones](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、利用可能な現在のリージョンのアベイラビリティゾーンについて記述します。

```
Get-EC2AvailabilityZone
```

出力:

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

例 2: この例では、障害状態にあるアベイラビリティゾーンを記述します。この例で 사용되는構文には、PowerShell バージョン 3 以降が必要です。

```
Get-EC2AvailabilityZone -Filter @{ Name="state";Values="impaired" }
```

例 3: PowerShell バージョン 2 では、New-Object を使用してフィルターを作成する必要があります。

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = "impaired"

Get-EC2AvailabilityZone -Filter $filter
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeAvailabilityZones](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、利用可能な現在のリージョンのアベイラビリティゾーンについて記述します。

```
Get-EC2AvailabilityZone
```

出力:

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

例 2: この例では、障害状態にあるアベイラビリティゾーンを記述します。この例で 사용되는構文には、PowerShell バージョン 3 以降が必要です。

```
Get-EC2AvailabilityZone -Filter @{ Name="state";Values="impaired" }
```

例 3: PowerShell バージョン 2 では、New-Object を使用してフィルターを作成する必要があります。

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = "impaired"

Get-EC2AvailabilityZone -Filter $filter
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeAvailabilityZones](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
```

```

        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

    def get_availability_zones(self) -> List[str]:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
        client.

```

```

: return: The list of Availability Zones for the client Region.
"""
try:
    response = self.ec2_client.describe_availability_zones()
    zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
    log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
except ClientError as err:
    log.error("Failed to retrieve availability zones.")
    log.error(f"Full error:\n\t{err}")
else:
    return zones

```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[DescribeAvailabilityZones](#)」を参照してください。

SAP ABAP

SDK for SAP ABAP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

TRY.
    oo_result = lo_ec2->describeavailabilityzones( ).
    " oo_result is returned for testing purposes. "
    DATA(lt_zones) = oo_result->get_availabilityzones( ).
    MESSAGE 'Retrieved information about Availability Zones.' TYPE 'I'.

    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception->av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.

```


- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「[DescribeAvailabilityZones](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeBundleTasks** を使用する

次のサンプルコードは、DescribeBundleTasks を使用方法を説明しています。

CLI

AWS CLI

バンドルタスクを記述するには

この例では、すべてのバンドルタスクを記述します。

コマンド:

```
aws ec2 describe-bundle-tasks
```

出力:

```
{
  "BundleTasks": [
    {
      "UpdateTime": "2015-09-15T13:26:54.000Z",
      "InstanceId": "i-1234567890abcdef0",
      "Storage": {
        "S3": {
          "Prefix": "winami",
          "Bucket": "bundletasks"
        }
      },
      "State": "bundling",
      "StartTime": "2015-09-15T13:24:35.000Z",
      "Progress": "3%",
      "BundleId": "bun-2a4e041c"
    }
  ]
}
```

```
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeBundleTasks](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定したバンドルタスクを記述します。

```
Get-EC2BundleTask -BundleId bun-12345678
```

例 2: この例では、状態が「完了」または「失敗」のバンドルタスクについて説明します。

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "complete", "failed" )

Get-EC2BundleTask -Filter $filter
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeBundleTasks](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定したバンドルタスクを記述します。

```
Get-EC2BundleTask -BundleId bun-12345678
```

例 2: この例では、状態が「完了」または「失敗」のバンドルタスクについて説明します。

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "complete", "failed" )

Get-EC2BundleTask -Filter $filter
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeBundleTasks](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeCapacityReservations** を使用する

次のサンプルコードは、DescribeCapacityReservations を使用する方法を説明しています。

CLI

AWS CLI

例 1: 1 つ以上のキャパシティ予約を記述するには

次の describe-capacity-reservations 例では、現在の AWS リージョンのすべてのキャパシティ予約の詳細を表示します。

```
aws ec2 describe-capacity-reservations
```

出力:

```
{
  "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": []
    }
]
}

```

例 2: 1 つ以上のキャパシティ予約を記述するには

次の `describe-capacity-reservations` の例では、指定したキャパシティ予約の詳細を表示します。

```

aws ec2 describe-capacity-reservations \
  --capacity-reservation-ids cr-1234abcd56EXAMPLE

```

出力:

```

{
  "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": []
    }
]
}

```

詳細については、「Amazon Elastic Compute Cloud User Guide for Linux Instances」の「[Viewing a Capacity Reservation](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeCapacityReservations](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、リージョンの 1 つ以上のキャパシティ予約について記述します。

```
Get-EC2CapacityReservation -Region eu-west-1
```

出力:

```

AvailabilityZone      : eu-west-1b
AvailableInstanceCount : 2
CapacityReservationId : cr-0c1f2345db6f7cdba

```

```

CreateDate       : 3/28/2019 9:29:41 AM
EbsOptimized     : True
EndDate         : 1/1/0001 12:00:00 AM
EndDateType      : unlimited
EphemeralStorage : False
InstanceMatchCriteria : open
InstancePlatform : Windows
InstanceType     : m4.xlarge
State           : active
Tags            : {}
Tenancy         : default
TotalInstanceCount : 2

```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeCapacityReservations](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、リージョンの 1 つ以上のキャパシティ予約について記述します。

```
Get-EC2CapacityReservation -Region eu-west-1
```

出力:

```

AvailabilityZone   : eu-west-1b
AvailableInstanceCount : 2
CapacityReservationId : cr-0c1f2345db6f7cdba
CreateDate        : 3/28/2019 9:29:41 AM
EbsOptimized      : True
EndDate          : 1/1/0001 12:00:00 AM
EndDateType       : unlimited
EphemeralStorage  : False
InstanceMatchCriteria : open
InstancePlatform  : Windows
InstanceType      : m4.xlarge
State            : active
Tags             : {}
Tenancy          : default
TotalInstanceCount : 2

```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeCapacityReservations](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeCustomerGateways** を使用する

次のサンプルコードは、DescribeCustomerGateways を使用方法を説明しています。

CLI

AWS CLI

カスタマーゲートウェイを記述するには

この例では、カスタマーゲートウェイを記述します。

コマンド:

```
aws ec2 describe-customer-gateways
```

出力:

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

特定のカスタマーゲートウェイを記述するには

この例では、指定されたカスタマーゲートウェイを記述します。

コマンド:

```
aws ec2 describe-customer-gateways --customer-gateway-ids cgw-0e11f167
```

出力:

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

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[DescribeCustomerGateways](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたカスタマーゲートウェイについて記述します。

```
Get-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

出力:

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

例 2: この例では、保留中または使用可能な状態のカスタマーゲートウェイを記述します。


```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2CustomerGateway -Filter $filter
```

例 3: この例では、すべてのカスタマーゲートウェイを記述します。

```
Get-EC2CustomerGateway
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeCustomerGateways](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたカスタマーゲートウェイについて記述します。

```
Get-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

出力:

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

例 2: この例では、保留中または使用可能な状態のカスタマーゲートウェイを記述します。

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2CustomerGateway -Filter $filter
```

例 3: この例では、すべてのカスタマーゲートウェイを記述します。

```
Get-EC2CustomerGateway
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeCustomerGateways](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeDhcpOptions** を使用する

次のサンプルコードは、DescribeDhcpOptions を使用する方法を説明しています。

CLI

AWS CLI

例 1: DHCP オプションを記述するには

次の describe-dhcp-options の例では、DHCP の詳細を取得します。

```
aws ec2 describe-dhcp-options
```

出力:

```
{
  "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"
  }
]
}

```

詳細については、「AWS VPC ユーザーガイド」の「[Working with DHCP Option Sets](#)」を参照してください。

例 2: DHCP オプションを記述し、出力をフィルタリングするには

次の describe-dhcp-options の例では、DHCP オプションを記述し、フィルターを使用して、ドメインネームサーバーに example.com を持つ DHCP オプションのみを返します。この例では、--query パラメータを使用して、出力に設定情報と ID のみを表示します。

```

aws ec2 describe-dhcp-options \
  --filters Name=key,Values=domain-name-servers Name=value,Values=example.com \
  --query "DhcpOptions[*].[DhcpConfigurations,DhcpOptionsId]"

```

出力:

```
[
  [
    [
      {
        "Key": "domain-name",
        "Values": [
          {
            "Value": "example.com"
          }
        ]
      },
      {
        "Key": "domain-name-servers",
        "Values": [
          {
            "Value": "172.16.16.16"
          }
        ]
      }
    ],
    "dopt-001122334455667ab"
  ]
]
```

詳細については、「AWS VPC ユーザーガイド」の「[Working with DHCP Option Sets](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeDhcpOptions](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、DHCP オプションセットを一覧表示します。

```
Get-EC2DhcpOption
```

出力:

DhcpConfigurations	DhcpOptionsId	Tag
-----	-----	---
{domain-name, domain-name-servers}	dopt-1a2b3c4d	{}
{domain-name, domain-name-servers}	dopt-2a3b4c5d	{}
{domain-name-servers}	dopt-3a4b5c6d	{}

例 2: この例では、指定された DHCP オプションセットの設定の詳細を取得します。

```
(Get-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d).DhcpConfigurations
```

出力:

Key	Values
---	-----
domain-name	{abc.local}
domain-name-servers	{10.0.0.101, 10.0.0.102}

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeDhcpOptions](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、DHCP オプションセットを一覧表示します。

```
Get-EC2DhcpOption
```

出力:

DhcpConfigurations	DhcpOptionsId	Tag
-----	-----	---
{domain-name, domain-name-servers}	dopt-1a2b3c4d	{}
{domain-name, domain-name-servers}	dopt-2a3b4c5d	{}
{domain-name-servers}	dopt-3a4b5c6d	{}

例 2: この例では、指定された DHCP オプションセットの設定の詳細を取得します。

```
(Get-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d).DhcpConfigurations
```

出力:

Key	Values

```
---
domain-name          {abc.local}
domain-name-servers  {10.0.0.101, 10.0.0.102}
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeDhcpOptions](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeFlowLogs** を使用する

次のサンプルコードは、DescribeFlowLogs を使用する方法を説明しています。

CLI

AWS CLI

例 1: すべてのフローログを記述するには

次の describe-flow-logs の例では、すべてのフローログの詳細を表示します。

```
aws ec2 describe-flow-logs
```

出力:

```
{
  "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}"
    }
]
}

```

例 2: フローログのサブセットを記述するには

次の describe-flow-logs の例では、フィルターを使用して、Amazon CloudWatch Logs の指定されたロググループに含まれるフローログの詳細のみを表示します。

```
aws ec2 describe-flow-logs \
    --filter "Name=log-group-name,Values=MyFlowLogs"
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeFlowLogs](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、ログの送信先タイプが「s3」の 1 つ以上のフローログを記述します。

```
Get-EC2FlowLog -Filter @{Name="log-destination-type";Values="s3"}
```

出力:

```

CreationTime           : 2/25/2019 9:07:36 PM
DeliverLogsErrorMessage :
DeliverLogsPermissionArn :
DeliverLogsStatus      : SUCCESS
FlowLogId              : f1-01b2e3d45f67f8901
FlowLogStatus          : ACTIVE
LogDestination          : arn:aws:s3:::amzn-s3-demo-bucket-dd-tata
LogDestinationType     : s3
LogGroupName           :
ResourceId              : eni-01d2dda3456b7e890
TrafficType             : ALL

```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeFlowLogs](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、ログの送信先タイプが「s3」の 1 つ以上のフローログを記述します。

```
Get-EC2FlowLog -Filter @{"Name"="log-destination-type";Values="s3"}
```

出力:

```

CreationTime           : 2/25/2019 9:07:36 PM
DeliverLogsErrorMessage :
DeliverLogsPermissionArn :
DeliverLogsStatus      : SUCCESS
FlowLogId              : f1-01b2e3d45f67f8901
FlowLogStatus          : ACTIVE
LogDestination          : arn:aws:s3:::amzn-s3-demo-bucket-dd-tata
LogDestinationType     : s3
LogGroupName           :
ResourceId              : eni-01d2dda3456b7e890
TrafficType             : ALL

```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeFlowLogs](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で `DescribeHostReservationOfferings` を使用する

次のサンプルコードは、`DescribeHostReservationOfferings` を使用方法を説明しています。

CLI

AWS CLI

専有ホスト予約サービスを記述するには

この例では、購入可能な M4 インスタンスファミリー専有ホストの予約を記述します。

コマンド:

```
aws ec2 describe-host-reservation-offerings --filter Name=instance-  
family,Values=m4
```

出力:

```
{  
  "OfferingSet": [  
    {  
      "HourlyPrice": "1.499",  
      "OfferingId": "hro-03f707bf363b6b324",  
      "InstanceFamily": "m4",  
      "PaymentOption": "NoUpfront",  
      "UpfrontPrice": "0.000",  
      "Duration": 31536000  
    },  
    {  
      "HourlyPrice": "1.045",  
      "OfferingId": "hro-0ef9181cabdef7a02",  
      "InstanceFamily": "m4",  
      "PaymentOption": "NoUpfront",  
      "UpfrontPrice": "0.000",  
      "Duration": 94608000  
    },  
    {  
      "HourlyPrice": "0.714",  
      "OfferingId": "hro-04567a15500b92a51",  
      "InstanceFamily": "m4",  
      "PaymentOption": "PartialUpfront",
```

```

        "UpfrontPrice": "6254.000",
        "Duration": 31536000
    },
    {
        "HourlyPrice": "0.484",
        "OfferingId": "hro-0d5d7a9d23ed7fbfe",
        "InstanceFamily": "m4",
        "PaymentOption": "PartialUpfront",
        "UpfrontPrice": "12720.000",
        "Duration": 94608000
    },
    {
        "HourlyPrice": "0.000",
        "OfferingId": "hro-05da4108ca998c2e5",
        "InstanceFamily": "m4",
        "PaymentOption": "AllUpfront",
        "UpfrontPrice": "23913.000",
        "Duration": 94608000
    },
    {
        "HourlyPrice": "0.000",
        "OfferingId": "hro-0a9f9be3b95a3dc8f",
        "InstanceFamily": "m4",
        "PaymentOption": "AllUpfront",
        "UpfrontPrice": "12257.000",
        "Duration": 31536000
    }
]
}

```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeHostReservationOfferings](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、PaymentOption が「NoUpfront」である、指定されたフィルター「instance-family」で購入できる専有ホスト予約を記述します。

```

Get-EC2HostReservationOffering -Filter @{Name="instance-family";Values="m4"} |
Where-Object PaymentOption -eq NoUpfront

```

出力:

```
CurrencyCode      :
Duration          : 94608000
HourlyPrice       : 1.307
InstanceFamily    : m4
OfferingId        : hro-0c1f234567890d9ab
PaymentOption     : NoUpfront
UpfrontPrice      : 0.000

CurrencyCode      :
Duration          : 31536000
HourlyPrice       : 1.830
InstanceFamily    : m4
OfferingId        : hro-04ad12aaaf34b5a67
PaymentOption     : NoUpfront
UpfrontPrice      : 0.000
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeHostReservationOfferings](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、PaymentOption が「NoUpfront」である、指定されたフィルター「instance-family」で購入できる専有ホスト予約を記述します。

```
Get-EC2HostReservationOffering -Filter @{"Name"="instance-family";Values="m4"} |
Where-Object PaymentOption -eq NoUpfront
```

出力:

```
CurrencyCode      :
Duration          : 94608000
HourlyPrice       : 1.307
InstanceFamily    : m4
OfferingId        : hro-0c1f234567890d9ab
PaymentOption     : NoUpfront
UpfrontPrice      : 0.000

CurrencyCode      :
Duration          : 31536000
HourlyPrice       : 1.830
```

```
InstanceFamily : m4
OfferingId      : hro-04ad12aaaf34b5a67
PaymentOption   : NoUpfront
UpfrontPrice    : 0.000
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeHostReservationOfferings](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeHosts** を使用する

次のサンプルコードは、DescribeHosts を使用する方法を説明しています。

CLI

AWS CLI

専有ホストの詳細を表示するには

次の describe-hosts 例では、AWS アカウント内の available Dedicated Hosts の詳細を表示します。

```
aws ec2 describe-hosts --filter "Name=state,Values=available"
```

出力:

```
{
  "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"
}
]
}

```

詳細については、「Amazon Elastic Compute Cloud User Guide for Linux Instances」の「[Viewing Dedicated Hosts](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeHosts](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、EC2 ホストの詳細を返します。

```
Get-EC2Host
```

出力:

```
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               : {}

```

例 2: この例では、ホスト h-01e23f4cd567899f1 の AvailableInstanceCapacity をクエリします。

```

Get-EC2Host -HostId h-01e23f4cd567899f1 | Select-Object -ExpandProperty
    AvailableCapacity | Select-Object -expand AvailableInstanceCapacity

```

出力:

```

AvailableCapacity InstanceType TotalCapacity
-----
11                m4.xlarge    11

```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeHosts](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、EC2 ホストの詳細を返します。

```

Get-EC2Host

```

出力:

```

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               : {}
```

例 2: この例では、ホスト h-01e23f4cd567899f1 の AvailableInstanceCapacity をクエリします。

```
Get-EC2Host -HostId h-01e23f4cd567899f1 | Select-Object -ExpandProperty
    AvailableCapacity | Select-Object -expand AvailableInstanceCapacity
```

出力:

```
AvailableCapacity InstanceType TotalCapacity
-----
11                m4.xlarge      11
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeHosts](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DescribeIamInstanceProfileAssociations**で を使用する

次のサンプルコードは、DescribeIamInstanceProfileAssociations を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [レジリエントなサービスの構築と管理](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Get the instance profile association data for an instance.
/// </summary>
/// <param name="instanceId">The Id of the instance.</param>
/// <returns>Instance profile associations data.</returns>
public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
{
    try
    {
        var response = await
        _amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
            new DescribeIamInstanceProfileAssociationsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new("instance-id", new List<string>() { instanceId })
                },
            });
        return response.IamInstanceProfileAssociations[0];
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
        {
            _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
        }

        throw;
    }
    catch (Exception ex)
```



```
{
    _logger.LogError(ex, $"An error occurred while creating the
template.: {ex.Message}");
    throw;
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[DescribeIamInstanceProfileAssociations](#)」を参照してください。

CLI

AWS CLI

IAM インスタンスプロファイルの関連付けについて説明するには

この例では、IAM インスタンスプロファイルのすべての関連付けについて説明しています。

コマンド:

```
aws ec2 describe-iam-instance-profile-associations
```

出力:

```
{
  "IamInstanceProfileAssociations": [
    {
      "InstanceId": "i-09eb09efa73ec1dee",
      "State": "associated",
      "AssociationId": "iip-assoc-0db249b1f25fa24b8",
      "IamInstanceProfile": {
        "Id": "AIPAJVQN4F5WVLGCJDRGM",
        "Arn": "arn:aws:iam::123456789012:instance-profile/admin-role"
      }
    },
    {
      "InstanceId": "i-0402909a2f4dffd14",
      "State": "associating",
      "AssociationId": "iip-assoc-0d1ec06278d29f44a",
      "IamInstanceProfile": {
        "Id": "AGJAJVQN4F5WVLGCJABCM",
        "Arn": "arn:aws:iam::123456789012:instance-profile/user1-role"
      }
    }
  ]
}
```

```
}  
  }  
]  
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeIamInstanceProfileAssociations](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
const ec2Client = new EC2Client({});  
const { IamInstanceProfileAssociations } = await ec2Client.send(  
  new DescribeIamInstanceProfileAssociationsCommand({  
    Filters: [  
      { Name: "instance-id", Values: [state.targetInstance.InstanceId] },  
    ],  
  }),  
);
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[DescribeIamInstanceProfileAssociations](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

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

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"

```

```

self.instance_role_name = f"{resource_prefix}-role"
self.instance_profile_name = f"{resource_prefix}-prof"

# Failure mode
self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
self.bad_creds_role_name = f"{resource_prefix}-bc-role"
self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
    """
    Gets data about the profile associated with an instance.

    :param instance_id: The ID of the instance to look up.
    :return: The profile data.
    """
    try:
        response =
self.ec2_client.describe_iam_instance_profile_associations(
            Filters=[{"Name": "instance-id", "Values": [instance_id]}]
        )
        if not response["IamInstanceProfileAssociations"]:
            log.info(f"No instance profile found for instance
{instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
            log.error(
                f"Failed to retrieve instance profile for instance
{instance_id}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")

```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[DescribeIamInstanceProfileAssociations](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で `DescribeIdFormat` を使用する

次のサンプルコードは、`DescribeIdFormat` を使用する方法を説明しています。

CLI

AWS CLI

例 1: リソースの ID 形式を記述するには

次の `describe-id-format` の例では、セキュリティグループの ID 形式について説明します。

```
aws ec2 describe-id-format \
  --resource security-group
```

次の出力例では、`Deadline` 値は、このリソースタイプが短い ID 形式から長い ID 形式に完全に切り替わる期限が、2018 年 8 月 15 日の 00:00 UTC に終了したことを示します。

```
{
  "Statuses": [
    {
      "Deadline": "2018-08-15T00:00:00.000Z",
      "Resource": "security-group",
      "UseLongIds": true
    }
  ]
}
```

例 2: すべてのリソースの ID 形式を記述するには

次の `describe-id-format` の例では、すべてのリソースタイプの ID 形式について説明します。短い ID 形式をサポートしたすべてのリソースタイプは、長い ID 形式を使用するように切り替えられました。

```
aws ec2 describe-id-format
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeIdFormat](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたリソースタイプの ID 形式を記述します。

```
Get-EC2IdFormat -Resource instance
```

出力:

Resource	UseLongIds
instance	False

例 2: この例では、長い ID をサポートするすべてのリソースタイプの ID 形式を記述します。

```
Get-EC2IdFormat
```

出力:

Resource	UseLongIds
reservation	False
instance	False

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeIdFormat](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたリソースタイプの ID 形式を記述します。

```
Get-EC2IdFormat -Resource instance
```

出力:

Resource	UseLongIds

```
-----
instance      False
```

例 2: この例では、長い ID をサポートするすべてのリソースタイプの ID 形式を記述します。

```
Get-EC2IdFormat
```

出力:

```
Resource      UseLongIds
-----
reservation    False
instance        False
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeIdFormat](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeIdentityIdFormat** を使用する

次のサンプルコードは、DescribeIdentityIdFormat を使用方法を説明しています。

CLI

AWS CLI

IAM ロールの ID 形式を記述するには

次のdescribe-identity-id-format例では、AWS アカウントの IAM ロールによって作成されたインスタンスEC2Roleによって受信される ID 形式について説明します。

```
aws ec2 describe-identity-id-format \
  --principal-arn arn:aws:iam::123456789012:role/my-iam-role \
  --resource instance
```

次の出力は、このロールによって作成されたインスタンスが長い ID 形式で ID を受け取ること示します。

```
{
  "Statuses": [
    {
      "Deadline": "2016-12-15T00:00:00Z",
      "Resource": "instance",
      "UseLongIds": true
    }
  ]
}
```

IAM ユーザーの ID 形式を記述するには

次のdescribe-identity-id-format例では、AdminUser AWS アカウントの IAM ユーザーが作成したスナップショットによって受信される ID 形式について説明します。

```
aws ec2 describe-identity-id-format \
  --principal-arn arn:aws:iam::123456789012:user/AdminUser \
  --resource snapshot
```

出力は、このユーザーによって作成されたスナップショットが長い ID 形式で ID を受け取ることを示します。

```
{
  "Statuses": [
    {
      "Deadline": "2016-12-15T00:00:00Z",
      "Resource": "snapshot",
      "UseLongIds": true
    }
  ]
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeIdentityIdFormat](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたロールのリソース「image」の ID 形式を返します。


```
Get-EC2IdentityIdFormat -PrincipalArn arn:aws:iam::123456789511:role/JDBC -
Resource image
```

出力:

```
Deadline          Resource UseLongIds
-----
8/2/2018 11:30:00 PM image      True
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeIdentityIdFormat](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたロールのリソース「image」の ID 形式を返します。

```
Get-EC2IdentityIdFormat -PrincipalArn arn:aws:iam::123456789511:role/JDBC -
Resource image
```

出力:

```
Deadline          Resource UseLongIds
-----
8/2/2018 11:30:00 PM image      True
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeIdentityIdFormat](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください [AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeImageAttribute** を使用する

次のサンプルコードは、DescribeImageAttribute を使用する方法を説明しています。

CLI

AWS CLI

AMI の起動許可を記述するには

この例では、指定された AMI の起動許可を記述します。

コマンド:

```
aws ec2 describe-image-attribute --image-id ami-5731123e --  
attribute LaunchPermission
```

出力:

```
{  
  "LaunchPermissions": [  
    {  
      "UserId": "123456789012"  
    }  
  ],  
  "ImageId": "ami-5731123e",  
}
```

AMI の製品コードを記述するには

この例では、指定された AMI の製品コードを記述します。この AMI には製品コードはありません。

コマンド:

```
aws ec2 describe-image-attribute --image-id ami-5731123e --attribute productCodes
```

出力:

```
{  
  "ProductCodes": [],  
  "ImageId": "ami-5731123e",  
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeImageAttribute](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定された AMI の説明を取得します。

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute description
```

出力:

```
BlockDeviceMappings : {}  
Description          : My image description  
ImageId              : ami-12345678  
KernelId             :  
LaunchPermissions    : {}  
ProductCodes         : {}  
RamdiskId            :  
SriovNetSupport       :
```

例 2: この例では、指定された AMI の起動許可を取得します。

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

出力:

```
BlockDeviceMappings : {}  
Description          :  
ImageId              : ami-12345678  
KernelId             :  
LaunchPermissions    : {all}  
ProductCodes         : {}  
RamdiskId            :  
SriovNetSupport       :
```

例 3: この例では、拡張ネットワーキングが有効になっているかどうかをテストします。

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute sriovNetSupport
```

出力:

```
BlockDeviceMappings : {}
Description          :
ImageId              : ami-12345678
KernelId             :
LaunchPermissions    : {}
ProductCodes         : {}
RamdiskId            :
SriovNetSupport      : simple
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeImageAttribute](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定された AMI の説明を取得します。

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute description
```

出力:

```
BlockDeviceMappings : {}
Description          : My image description
ImageId              : ami-12345678
KernelId             :
LaunchPermissions    : {}
ProductCodes         : {}
RamdiskId            :
SriovNetSupport      :
```

例 2: この例では、指定された AMI の起動許可を取得します。

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

出力:

```
BlockDeviceMappings : {}
Description          :
ImageId              : ami-12345678
KernelId             :
LaunchPermissions    : {all}
```

```
ProductCodes      : {}  
RamdiskId         :  
SriovNetSupport   :
```

例 3: この例では、拡張ネットワーキングが有効になっているかどうかをテストします。

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute sriovNetSupport
```

出力:

```
BlockDeviceMappings : {}  
Description         :  
ImageId             : ami-12345678  
KernelId            :  
LaunchPermissions   : {}  
ProductCodes        : {}  
RamdiskId           :  
SriovNetSupport     : simple
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeImageAttribute](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DescribeImages**で を使用する

次のサンプルコードは、DescribeImages を使用方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)
- [プライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成する](#)
- [Amazon VPC の使用を開始する](#)

Bash

AWS CLI Bash スクリプトを使用する

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

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

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

```

        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
}

```

この例で使用されているユーティリティ関数。

```

#####
# 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
}

```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[DescribeImages](#)」を参照してください。

CLI

AWS CLI

例 1: AMI を説明するには

次の `describe-images` の例では、指定された AMI 内のインターフェイスについて説明します。

```
aws ec2 describe-images \  
  --region us-east-1 \  
  --image-ids ami-1234567890EXAMPLE
```

出力:

```
{  
  "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"
    }
]
}

```

詳細については「アマゾン EC2 ユーザーガイド」の「[アマゾン マシンイメージ \(AMI\)](#)」を参照してください。

例 2: フィルターに基づいて AMI を説明するには

次の describe-images の例では、Amazon が提供する、Amazon EBS を基にした Windows AMI を説明しています。

```

aws ec2 describe-images \
  --owners amazon \
  --filters "Name=platform,Values=windows" "Name=root-device-type,Values=ebs"

```

describe-images の出力例については、例 1 を参照してください。

フィルターを使用するその他の例については、「Amazon EC2 ユーザーガイド」の「[リソースの一覧表示とフィルタリングの方法](#)」を参照してください。

例 3: タグに基づいて AMI を説明するには

次の describe-images の例では、タグ Type=Custom が付いたすべての AMI について説明しています。この例では、--query パラメータを使用して AMI ID のみを表示します。

```

aws ec2 describe-images \
  --filters "Name=tag:Type,Values=Custom" \
  --query 'Images[*].[ImageId]' \
  --output text

```

出力:

```

ami-1234567890EXAMPLE
ami-0abcdef1234567890

```

タグフィルターを使用するその他の例については、「Amazon EC2 ユーザーガイド」の「[タグの使用方法](#)」を参照してください。

- API の詳細については、「AWS CLI Command Reference」の「[DescribeImages](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

/**
 * Describes the specified images (AMIs, AKIs, and ARIs) available to you or all
 * of the images available to you.
 * @param {{ architecture: string, pageSize: number }} options
 */
export const main = async ({ architecture, pageSize }) => {
  pageSize = Number.parseInt(pageSize);
  const client = new EC2Client({});

  // The paginate function is a wrapper around the base command.
  const paginator = paginateDescribeImages(
    // Without limiting the page size, this call can take a long time. pageSize
    // is just sugar for
    // the MaxResults property in the base command.
    { client, pageSize },
    {
      // There are almost 70,000 images available. Be specific with your
      // filtering
      // to increase efficiency.
      // See https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/clients/
      // client-ec2/interfaces/describeimagescommandinput.html#filters
      Filters: [{ Name: "architecture", Values: [architecture] }],
    },
  );
};
```

```
/**
 * @type {import('@aws-sdk/client-ec2').Image[]}
 */
const images = [];
let recordsScanned = 0;

try {
  for await (const page of paginator) {
    recordsScanned += pageSize;
    if (page.Images.length) {
      images.push(...page.Images);
      break;
    }
    console.log(
      `No matching image found yet. Searched ${recordsScanned} records.`,
    );
  }

  if (images.length) {
    console.log(
      `Found ${images.length} images:\n\n${images.map((image) =>
image.Name).join("\n")}\n`,
    );
  } else {
    console.log(
      `No matching images found. Searched ${recordsScanned} records.\n`,
    );
  }

  return images;
} catch (caught) {
  if (caught instanceof Error && caught.name === "InvalidParameterValue") {
    console.warn(`${caught.message}`);
    return [];
  }
  throw caught;
}
};
```

- API の詳細については、AWS SDK for JavaScript API リファレンスの「[DescribeImages](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定された AMI を記述します。

```
Get-EC2Image -ImageId ami-12345678
```

出力:

```
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
```

例 2: この例では、所有している AMI を記述します。

```
Get-EC2Image -owner self
```

例 3: この例では、Microsoft Windows Server を実行するパブリック AMI を記述します。

```
Get-EC2Image -Filter @{ Name="platform"; Values="windows" }
```

例 4: この例では、「us-west-2」リージョンのすべてのパブリック AMI を記述します。

```
Get-EC2Image -Region us-west-2
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeImages](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定された AMI を記述します。

```
Get-EC2Image -ImageId ami-12345678
```

出力:

```
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
```

例 2: この例では、所有している AMI を記述します。

```
Get-EC2Image -owner self
```

例 3: この例では、Microsoft Windows Server を実行するパブリック AMI を記述します。

```
Get-EC2Image -Filter @{ Name="platform"; Values="windows" }
```

例 4: この例では、「us-west-2」リージョンのすべてのパブリック AMI を記述します。

```
Get-EC2Image -Region us-west-2
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeImages](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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

```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[DescribeImages](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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)
    }
}
```

SSM で `list_images` 関数を使用して、環境に基づいて制限します。SSM の詳細については、https://docs.aws.amazon.com/systems-manager/latest/userguide/example_ssm_GetParameters_section.html を参照してください。

```
async fn find_image(&mut self) -> Result<ScenarioImage, EC2Error> {
    let params: Vec<Parameter> = self
        .ssm
        .list_path("/aws/service/ami-amazon-linux-latest")
        .await
        .map_err(|e| e.add_message("Could not find parameters for available
images"))?
        .into_iter()
```

```

        .filter(|param| param.name().is_some_and(|name|
name.contains("amzn2"))))
        .collect();
    let amzn2_images: Vec<ScenarioImage> = self
        .ec2
        .list_images(params)
        .await
        .map_err(|e| e.add_message("Could not find images"))?
        .into_iter()
        .map(ScenarioImage::from)
        .collect();
    println!("We will now create an instance from an Amazon Linux 2 AMI");
    let ami = self.util.select_scenario_image(amzn2_images)?;
    Ok(ami)
}

```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[DescribeImages](#)」を参照してください。

SAP ABAP

SDK for SAP ABAP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

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.

```

- API の詳細については、AWS SDK for SAP ABAP API リファレンスの「[DescribeImages](#)」を参照してください。

Swift

SDK for Swift

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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 []
}
}

```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[DescribeImages](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeImportImageTasks** を使用する

次のサンプルコードは、DescribeImportImageTasks を使用方法を説明しています。

CLI

AWS CLI

イメージのインポートタスクをモニタリングするには

次の describe-import-image-tasks の例では、指定したイメージのインポートタスクをチェックします。

```
aws ec2 describe-import-image-tasks \
  --import-task-ids import-ami-1234567890abcdef0
```

進行中のイメージのインポートタスクの出力。

```
{
  "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"
    }
]
}

```

完了したイメージのインポートタスクの出力。結果の AMI の ID は ImageId によって指定されます。

```

{
    "ImportImageTasks": [
        {
            "ImportTaskId": "import-ami-1234567890abcdef0",
            "ImageId": "ami-1234567890abcdef0",
            "SnapshotDetails": [
                {
                    "DiskImageSize": 705638400.0,
                    "Format": "ova",
                    "SnapshotId": "snap-1234567890abcdef0"
                    "Status": "completed",
                    "UserBucket": {
                        "S3Bucket": "my-import-bucket",
                        "S3Key": "vms/my-server-vm.ova"
                    }
                }
            ],
            "Status": "completed"
        }
    ]
}

```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeImportImageTasks](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたイメージのインポートタスクを記述します。

```
Get-EC2ImportImageTask -ImportTaskId import-ami-hgfedcba
```

出力:

```
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     :
```

例 2: この例では、すべてのイメージのインポートタスクを記述します。

```
Get-EC2ImportImageTask
```

出力:

```
Architecture      :
Description       : Windows Image 1
Hypervisor        :
ImageId           :
ImportTaskId      : import-ami-abcdefgh
LicenseType       : AWS
Platform          : Windows
Progress          :
SnapshotDetails   : {}
```

```
Status      : deleted
StatusMessage : User initiated task cancelation

Architecture : x86_64
Description   : Windows Image 2
Hypervisor    :
ImageId       : ami-1a2b3c4d
ImportTaskId  : import-ami-hgfedcba
LicenseType   : AWS
Platform      : Windows
Progress      :
SnapshotDetails : {/dev/sda1}
Status        : completed
StatusMessage :
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeImportImageTasks](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたイメージのインポートタスクを記述します。

```
Get-EC2ImportImageTask -ImportTaskId import-ami-hgfedcba
```

出力:

```
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 :
```

例 2: この例では、すべてのイメージのインポートタスクを記述します。

```
Get-EC2ImportImageTask
```

出力:

```
Architecture      :  
Description       : Windows Image 1  
Hypervisor        :  
ImageId           :  
ImportTaskId      : import-ami-abcdefgh  
LicenseType       : AWS  
Platform          : Windows  
Progress          :  
SnapshotDetails   : {}  
Status            : deleted  
StatusMessage     : User initiated task cancelation  
  
Architecture      : x86_64  
Description       : Windows Image 2  
Hypervisor        :  
ImageId           : ami-1a2b3c4d  
ImportTaskId      : import-ami-hgfedcba  
LicenseType       : AWS  
Platform          : Windows  
Progress          :  
SnapshotDetails   : {/dev/sda1}  
Status            : completed  
StatusMessage     :
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeImportImageTasks](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeImportSnapshotTasks** を使用する

次のサンプルコードは、DescribeImportSnapshotTasks を使用する方法を説明しています。

CLI

AWS CLI

スナップショットのインポートタスクをモニタリングするには

次の `describe-import-snapshot-tasks` の例では、指定したインポートスナップショットタスクをチェックします。

```
aws ec2 describe-import-snapshot-tasks \
  --import-task-ids import-snap-1234567890abcdef0
```

進行中のスナップショットのインポートタスクの出力:

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

完了したスナップショットのインポートタスクの出力。結果のスナップショットの ID は `SnapshotId` によって指定されます。

```
{
  "ImportSnapshotTasks": [
    {
      "Description": "My server VMDK",
      "ImportTaskId": "import-snap-1234567890abcdef0",
      "SnapshotTaskDetail": {
        "Description": "My server VMDK",
        "DiskImageSize": "705638400.0",
        "Format": "VMDK",
        "SnapshotId": "snap-1234567890abcdef0"
      }
    }
  ]
}
```

```

        "Status": "completed",
        "UserBucket": {
            "S3Bucket": "my-import-bucket",
            "S3Key": "vms/my-server-vm.vmdk"
        }
    }
}
]
}

```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeImportSnapshotTasks](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたスナップショットのインポートタスクを記述します。

```
Get-EC2ImportSnapshotTask -ImportTaskId import-snap-abcdefgh
```

出力:

Description	ImportTaskId	SnapshotTaskDetail
-----	-----	-----
Disk Image Import 1	import-snap-abcdefgh	
Amazon.EC2.Model.SnapshotTaskDetail		

例 2: この例では、すべてのスナップショットのインポートタスクを記述します。

```
Get-EC2ImportSnapshotTask
```

出力:

Description	ImportTaskId	SnapshotTaskDetail
-----	-----	-----

```
Disk Image Import 1      import-snap-abcdefgh
Amazon.EC2.Model.SnapshotTaskDetail
Disk Image Import 2      import-snap-hgfedcba
Amazon.EC2.Model.SnapshotTaskDetail
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeImportSnapshotTasks](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたスナップショットのインポートタスクを記述します。

```
Get-EC2ImportSnapshotTask -ImportTaskId import-snap-abcdefgh
```

出力:

Description	ImportTaskId	SnapshotTaskDetail
-----	-----	-----
Disk Image Import 1 Amazon.EC2.Model.SnapshotTaskDetail	import-snap-abcdefgh	

例 2: この例では、すべてのスナップショットのインポートタスクを記述します。

```
Get-EC2ImportSnapshotTask
```

出力:

Description	ImportTaskId	SnapshotTaskDetail
-----	-----	-----
Disk Image Import 1 Amazon.EC2.Model.SnapshotTaskDetail	import-snap-abcdefgh	
Disk Image Import 2 Amazon.EC2.Model.SnapshotTaskDetail	import-snap-hgfedcba	

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeImportSnapshotTasks](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で `DescribeInstanceAttribute` を使用する

次のサンプルコードは、`DescribeInstanceAttribute` を使用する方法を説明しています。

CLI

AWS CLI

インスタンスタイプを記述するには

この例では、指定されたインスタンスのインスタンスタイプを記述します。

コマンド:

```
aws ec2 describe-instance-attribute --instance-id i-1234567890abcdef0 --  
attribute instanceType
```

出力:

```
{  
  "InstanceId": "i-1234567890abcdef0"  
  "InstanceType": {  
    "Value": "t1.micro"  
  }  
}
```

`disableApiTermination` 属性を記述するには

この例では、指定されたインスタンスの `disableApiTermination` 属性を記述します。

コマンド:

```
aws ec2 describe-instance-attribute --instance-id i-1234567890abcdef0 --  
attribute disableApiTermination
```

出力:

```
{
```

```
"InstanceId": "i-1234567890abcdef0"
  "DisableApiTermination": {
    "Value": "false"
  }
}
```

インスタンスのブロックデバイスマッピングを記述するには

この例では、指定されたインスタンスの `blockDeviceMapping` 属性を記述します。

コマンド:

```
aws ec2 describe-instance-attribute --instance-id i-1234567890abcdef0 --
attribute blockDeviceMapping
```

出力:

```
{
  "InstanceId": "i-1234567890abcdef0"
  "BlockDeviceMappings": [
    {
      "DeviceName": "/dev/sda1",
      "Ebs": {
        "Status": "attached",
        "DeleteOnTermination": true,
        "VolumeId": "vol-049df61146c4d7901",
        "AttachTime": "2013-05-17T22:42:34.000Z"
      }
    },
    {
      "DeviceName": "/dev/sdf",
      "Ebs": {
        "Status": "attached",
        "DeleteOnTermination": false,
        "VolumeId": "vol-049df61146c4d7901",
        "AttachTime": "2013-09-10T23:07:00.000Z"
      }
    }
  ],
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeInstanceAttribute](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたインスタンスのインスタンスタイプを記述します。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute instanceType
```

出力:

```
InstanceType          : t2.micro
```

例 2: この例では、指定されたインスタンスで拡張ネットワーキングが有効になっているかどうかを記述します。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

出力:

```
SriovNetSupport        : simple
```

例 3: この例では、指定されたインスタンスのセキュリティグループについて記述します。

```
(Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute groupSet).Groups
```

出力:

```
GroupId
-----
sg-12345678
sg-45678901
```

例 4: この例では、指定したインスタンスで EBS 最適化が有効になっているかどうかを記述します。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

出力:

```
EbsOptimized                : False
```

例 5: この例では、指定されたインスタンスの「disableApiTermination」属性を記述します。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

出力:

```
DisableApiTermination       : False
```

例 6: この例では、指定されたインスタンスの「instanceInitiatedShutdownBehavior」属性を記述します。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute  
instanceInitiatedShutdownBehavior
```

出力:

```
InstanceInitiatedShutdownBehavior : stop
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeInstanceAttribute](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたインスタンスのインスタンスタイプを記述します。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute instanceType
```

出力:

```
InstanceType                : t2.micro
```

例 2: この例では、指定されたインスタンスで拡張ネットワーキングが有効になっているかどうかを記述します。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

出力:

```
SriovNetSupport          : simple
```

例 3: この例では、指定されたインスタンスのセキュリティグループについて記述します。

```
(Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute groupSet).Groups
```

出力:

```
GroupId
-----
sg-12345678
sg-45678901
```

例 4: この例では、指定したインスタンスで EBS 最適化が有効になっているかどうかを記述します。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

出力:

```
EbsOptimized          : False
```

例 5: この例では、指定されたインスタンスの「disableApiTermination」属性を記述します。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

出力:

```
DisableApiTermination : False
```

例 6: この例では、指定されたインスタンスの「instanceInitiatedShutdownBehavior」属性を記述します。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
instanceInitiatedShutdownBehavior
```

出力:

```
InstanceInitiatedShutdownBehavior : stop
```


- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeInstanceAttribute](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DescribeInstanceStatus**で を使用する

次のサンプルコードは、DescribeInstanceStatus を使用する方法を説明しています。

CLI

AWS CLI

インスタンスのステータスを表示するには

次の describe-instance-status の例では、指定したインスタンスの現在のステータスを示しています。

```
aws ec2 describe-instance-status \  
  --instance-ids i-1234567890abcdef0
```

出力:

```
{  
  "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"
        }
      ]
    }
  ]
}

```

詳細については、「Amazon EC2 ユーザーガイド」の「[インスタンスのステータスのモニタリング](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeInstanceStatus](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたインスタンスのステータスを記述します。

```
Get-EC2InstanceStatus -InstanceId i-12345678
```

出力:

```

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

```

出力:

Code	Name
----	----
16	running

```
$status.Status
```

出力:

Details	Status
-----	-----
{reachability}	ok

```
$status.SystemStatus
```

出力:

Details	Status
-----	-----
{reachability}	ok

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeInstanceStatus](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたインスタンスのステータスを記述します。

```
Get-EC2InstanceStatus -InstanceId i-12345678
```

出力:

```
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
```

出力:

Code	Name
----	----
16	running

```
$status.Status
```

出力:

Details	Status
-----	-----
{reachability}	ok

```
$status.SystemStatus
```

出力:

Details	Status
-----	-----
{reachability}	ok

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeInstanceStatus](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
async fn show_all_events(client: &Client) -> Result<(), Error> {  
    let resp = client.describe_regions().send().await.unwrap();
```

```

    for region in resp.regions.unwrap_or_default() {
        let reg: &'static str =
Box::leak(Box::from(region.region_name().unwrap()));
        let region_provider =
RegionProviderChain::default_provider().or_else(reg);
        let config = aws_config::from_env().region(region_provider).load().await;
        let new_client = Client::new(&config);

        let resp = new_client.describe_instance_status().send().await;

        println!("Instances in region {}: ", reg);
        println!();

        for status in resp.unwrap().instance_statuses() {
            println!(
                "    Events scheduled for instance ID: {}",
                status.instance_id().unwrap_or_default()
            );
            for event in status.events() {
                println!("        Event ID:      {}",
event.instance_event_id().unwrap());
                println!("        Description:  {}", event.description().unwrap());
                println!("        Event code:   {}", event.code().unwrap().as_ref());
                println!();
            }
        }
    }

    Ok(())
}

```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[DescribeInstanceStatus](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DescribeInstanceTypes**で を使用する

次のサンプルコードは、DescribeInstanceTypes を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

.NET

SDK for .NET

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

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

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

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

```

    }

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

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

```

- APIの詳細については、AWS SDK for .NET API リファレンスの「[DescribeInstanceTypes](#)」を参照してください。

Bash

AWS CLI Bash スクリプトを使用する

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

#####
# 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
}

```



```

    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
}

```

この例で使用されているユーティリティ関数。

```

#####
# 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
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeInstanceTypes](#)」を参照してください。

CLI

AWS CLI

例 1: インスタンスタイプを説明するには

次の describe-instance-types の例では、指定されたインスタンスタイプの詳細を表示します。

```
aws ec2 describe-instance-types \
  --instance-types t2.micro
```

出力:

```
{
  "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
}
]
}
```

詳細については、「Amazon Elastic Compute Cloud User Guide for Linux Instances」の「[Instance Types](#)」を参照してください。

例 2: 使用可能なインスタンスタイプをフィルタリングするには

フィルターを指定して、特定の特性を持つインスタンスタイプに結果を絞り込みます。次の `describe-instance-types` の例では、休止状態をサポートするインスタンスタイプを一覧表示しています。

```
aws ec2 describe-instance-types \
  --filters Name=hibernation-supported,Values=true --query
  'InstanceTypes[*].InstanceType'
```

出力:

```
[
  "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"
]

```

詳細については、「Amazon Elastic Compute Cloud User Guide for Linux Instances」の「[Instance Types](#)」を参照してください。

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[DescribeInstanceTypes](#)」を参照してください。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

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

```

```
    *
    * @return a {@link CompletableFuture} that resolves to the instance type
    string for the
    * "t2.2xlarge" instance type, or an empty string if the instance type is not
    found
    */
    public CompletableFuture<String> getInstanceTypesAsync() {
        DescribeInstanceTypesRequest typesRequest =
        DescribeInstanceTypesRequest.builder()
            .maxResults(10)
            .build();

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

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

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[DescribeInstanceTypes](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

/**
 * Describes the specified instance types. By default, all instance types for the
 * current Region are described. Alternatively, you can filter the results.
 * @param {{ pageSize: string, supportedArch: string[], freeTier: boolean }}
 * options
 */
export const main = async ({ pageSize, supportedArch, freeTier }) => {
  pageSize = Number.parseInt(pageSize);
  const client = new EC2Client({});

  // The paginate function is a wrapper around the underlying command.
  const paginator = paginateDescribeInstanceTypes(
    // Without limiting the page size, this call can take a long time. pageSize
    // is just sugar for
    // the MaxResults property in the underlying command.
    { client, pageSize },
    {
      Filters: [
        {
          Name: "processor-info.supported-architecture",
          Values: supportedArch,
        },
        { Name: "free-tier-eligible", Values: [freeTier ? "true" : "false"] },
      ],
    },
  );
};
```



```
try {
  /**
   * @type {import('@aws-sdk/client-ec2').InstanceTypeInfo[]}
   */
  const instanceTypes = [];

  for await (const page of paginator) {
    if (page.InstanceTypes.length) {
      instanceTypes.push(...page.InstanceTypes);

      // When we have at least 1 result, we can stop.
      if (instanceTypes.length >= 1) {
        break;
      }
    }
  }
  console.log(
    `Memory size in MiB for matching instance types:\n\n
    ${instanceTypes.map((it) => `${it.InstanceType}: ${it.MemoryInfo.SizeInMiB}
    MiB`).join("\n")}`,
  );
} catch (caught) {
  if (caught instanceof Error && caught.name === "InvalidParameterValue") {
    console.warn(`${caught.message}`);
    return [];
  }
  throw caught;
}
};
```

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の「[DescribeInstanceTypes](#)」を参照してください。

Kotlin

SDK for Kotlin

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
// 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
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[DescribeInstanceTypes](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
    using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
        instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
        level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
        objects. These are high-level objects that
                           wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
        EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)
```

```

def get_instance_types(
    self, architecture: str = "x86_64", sizes: List[str] = ["*.micro",
    "*.small"]
) -> List[Dict[str, Any]]:
    """
    Gets instance types that support the specified architecture and size.
    See https://docs.aws.amazon.com/AWSEC2/latest/APIReference/
    API_DescribeInstanceTypes.html
    for a list of allowable parameters.

    :param architecture: The architecture supported by instance types.
    Default: 'x86_64'.
    :param sizes: The size of instance types. Default: '*.micro', '*.small',
    :return: A list of dictionaries representing instance types that support
    the specified architecture and size.
    """
    try:
        inst_types = []
        paginator = self.ec2_client.get_paginator("describe_instance_types")
        for page in paginator.paginate(
            Filters=[
                {
                    "Name": "processor-info.supported-architecture",
                    "Values": [architecture],
                },
                {"Name": "instance-type", "Values": sizes},
            ]
        ):
            inst_types += page["InstanceTypes"]
    except ClientError as err:
        logger.error(
            f"Failed to get instance types: {architecture},
            {'.'.join(map(str, sizes))}"
        )
        error_code = err.response["Error"]["Code"]
        if error_code == "InvalidParameterValue":
            logger.error(
                "Parameters are invalid. "
                "Ensure architecture and size strings conform to
                DescribeInstanceTypes API reference."
            )
            raise
    else:

```

```
return inst_types
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[DescribeInstanceTypes](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// List instance types that match an image's architecture and are free tier
eligible.
pub async fn list_instance_types(&self, image: &Image) ->
Result<Vec<InstanceType>, EC2Error> {
    let architecture = format!(
        "{}",
        image.architecture().ok_or_else(|| EC2Error::new(format!(
            "Image {:?} does not have a listed architecture",
            image.image_id()
        )))?
    );
    let free_tier_eligible_filter = Filter::builder()
        .name("free-tier-eligible")
        .values("false")
        .build();
    let supported_architecture_filter = Filter::builder()
        .name("processor-info.supported-architecture")
        .values(architecture)
        .build();
    let response = self
        .client
        .describe_instance_types()
        .filters(free_tier_eligible_filter)
        .filters(supported_architecture_filter)
```

```

        .send()
        .await?;

    Ok(response
        .instance_types
        .unwrap_or_default()
        .into_iter()
        .filter_map(|iti| iti.instance_type)
        .collect())
}

```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[DescribeInstanceTypes](#)」を参照してください。

SAP ABAP

SDK for SAP ABAP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

" 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.

```

- APIの詳細については、AWS SDK for SAP ABAP API リファレンスの「[DescribeInstanceTypes](#)」を参照してください。

Swift

SDK for Swift

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

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
}
```

- APIの詳細については、「AWS SDK for Swift API リファレンス」の「[DescribeInstanceTypes](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DescribeInstances**で を使用する

次のサンプルコードは、DescribeInstances を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)
- [レジリエントなサービスの構築と管理](#)
- [Amazon VPC の使用を開始する](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

- API の詳細については、AWS SDK for .NET API リファレンスの「[DescribeInstances](#)」を参照してください。

Bash

AWS CLI Bash スクリプトを使用する

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

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

```

}

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

local aws_cli_args=()

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

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

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

```

```

    echo "$response"

    return 0
}

```

この例で使用されているユーティリティ関数。

```

#####
# 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."
    fi
}

```

```

elif [ "$err_code" == 254 ]; then
    errecho " The service returned an error."
elif [ "$err_code" == 255 ]; then
    errecho " 255 is a catch-all error."
fi

return 0
}

```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[DescribeInstances](#)」を参照してください。

C++

SDK for C++

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

//! Describe all Amazon Elastic Compute Cloud (Amazon EC2) instances associated
with an account.
/*!
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::EC2::describeInstances(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeInstancesRequest request;
    bool header = false;
    bool done = false;
    while (!done) {
        Aws::EC2::Model::DescribeInstancesOutcome outcome =
ec2Client.DescribeInstances(request);
        if (outcome.IsSuccess()) {
            if (!header) {
                std::cout << std::left <<

```

```

        std::setw(48) << "Name" <<
        std::setw(20) << "ID" <<
        std::setw(25) << "Ami" <<
        std::setw(15) << "Type" <<
        std::setw(15) << "State" <<
        std::setw(15) << "Monitoring" << std::endl;
    header = true;
}

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

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

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

        Aws::String typeString =

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

        Aws::String monitorString =

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

        const std::vector<Aws::EC2::Model::Tag> &tags =
instance.GetTags();
        auto nameIter = std::find_if(tags.cbegin(), tags.cend(),
            [](const Aws::EC2::Model::Tag
&tag) {
                return tag.GetKey() ==
                "Name";
            });
        if (nameIter != tags.cend()) {
            name = nameIter->GetValue();
        }
        std::cout <<

```

```

        std::setw(48) << name <<
        std::setw(20) << instance.GetInstanceId() <<
        std::setw(25) << instance.GetImageId() <<
        std::setw(15) << typeString <<
        std::setw(15) << instanceStateString <<
        std::setw(15) << monitorString << std::endl;
    }
}

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

return true;
}

```

- APIの詳細については、AWS SDK for C++ API リファレンスの「[DescribeInstances](#)」を参照してください。

CLI

AWS CLI

例 1: インスタンスを説明するには

次の describe-instances の例では、指定したインスタンスを示しています。

```
aws ec2 describe-instances \
    --instance-ids i-1234567890abcdef0
```

出力:

```
{
```



```

"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"
}
]
}

```

例 2: 指定したタイプのインスタンスをフィルタリングするには

次の `describe-instances` の例では、フィルターを使用して、指定されたタイプのインスタンスに結果の範囲を限定しています。

```
aws ec2 describe-instances \
    --filters Name=instance-type,Values=m5.large
```

出力例については、例 1 を参照してください。

詳細については、「Amazon EC2 ユーザーガイド」の「[CLI を使用した一覧表示およびフィルタリング](#)」を参照してください。

例 3: 指定したタイプとアベイラビリティーゾーンでインスタンスをフィルタリングするには

次の `describe-instances` の例では、複数のフィルターを使用して、指定されたアベイラビリティゾーンにある、指定されたタイプのインスタンスに結果を絞り込みます。

```
aws ec2 describe-instances \  
  --filters Name=instance-type,Values=t2.micro,t3.micro Name=availability-  
zone,Values=us-east-2c
```

出力例については、例 1 を参照してください。

例 4: JSON ファイルを使用して、指定したタイプとアベイラビリティゾーンでインスタンスをフィルタリングするには

次の `describe-instances` の例では、JSON 入力ファイルを使用して、前の例と同じフィルタリングを実行します。フィルターが複雑になるほど、JSON ファイル内での指定が簡単になります。

```
aws ec2 describe-instances \  
  --filters file://filters.json
```

`filters.json` の内容:

```
[  
  {  
    "Name": "instance-type",  
    "Values": ["t2.micro", "t3.micro"]  
  },  
  {  
    "Name": "availability-zone",  
    "Values": ["us-east-2c"]  
  }  
]
```

出力例については、例 1 を参照してください。

例 5: 指定した Owner タグを持つインスタンスをフィルタリングするには

次の `describe-instances` の例では、タグフィルターを使用して、タグ値に関係なく、指定されたタグキー (Owner) のタグを持つインスタンスに結果を絞り込みます。

```
aws ec2 describe-instances \  
  --filters Key=Owner
```

```
--filters "Name=tag-key,Values=Owner"
```

出力例については、例 1 を参照してください。

例 6: 指定した my-team タグ値を持つインスタンスをフィルタリングするには

次の describe-instances の例では、タグフィルターを使用して、タグキーに関係なく、指定されたタグ値 (my-team) のタグを持つインスタンスに結果を絞り込みます。

```
aws ec2 describe-instances \  
  --filters "Name=tag-value,Values=my-team"
```

出力例については、例 1 を参照してください。

例 7: 指定した Owner タグと my-team 値を持つインスタンスをフィルタリングするには

次の describe-instances の例では、タグフィルターを使用して、指定したタグ (Owner=my-team) を持つインスタンスに結果を絞り込みます。

```
aws ec2 describe-instances \  
  --filters "Name=tag:Owner,Values=my-team"
```

出力例については、例 1 を参照してください。

例 8: すべてのインスタンスのインスタンス ID とサブネット ID のみを表示するには

次の describe-instances の例では、--query パラメータを使用して、すべてのインスタンスのインスタンス ID とサブネット ID のみを JSON 形式で表示します。

Linux および 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
```

出力:

```
[
  {
    "Instance": "i-057750d42936e468a",
    "Subnet": "subnet-069beee9b12030077"
  },
  {
    "Instance": "i-001efd250faaa6ffa",
    "Subnet": "subnet-0b715c6b7db68927a"
  },
  {
    "Instance": "i-027552a73f021f3bd",
    "Subnet": "subnet-0250c25a1f4e15235"
  }
  ...
]
```

例 9: 指定したタイプのインスタンスをフィルタリングし、そのインスタンス ID のみを表示するには

次の `describe-instances` の例では、フィルターを使用して、指定されたタイプのインスタンスに結果を絞り込み、`--query` パラメータを使用してインスタンス ID のみを表示します。

```
aws ec2 describe-instances \
  --filters "Name=instance-type,Values=t2.micro" \
  --query "Reservations[*].Instances[*].[InstanceId]" \
  --output text
```

出力:

```
i-031c0dc19de2fb70c
i-00d8bfff789a736b75
i-0b715c6b7db68927a
i-0626d4edd54f1286d
i-00b8ae04f9f99908e
i-0fc71c25d2374130c
```

例 10: 指定したタイプのインスタンスをフィルタリングし、インスタンス ID、アベイラビリティゾーン、指定したタグ値のみを表示するには

次の describe-instances の例では、tag-key という名前のタグを持つインスタンスのインスタンス ID、アベイラビリティゾーン、および Name タグの値を表形式で表示します。

Linux および 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
```

出力:

```
-----
|                               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   |
+-----+-----+-----+
```

例 11: パーティションプレースメントグループ内のインスタンスを説明するには

次の describe-instances の例では、指定したインスタンスを示しています。出力にはインスタンスのプレースメント情報が含まれています。この情報にはインスタンスのプレースメントグループ名とパーティション番号が含まれます。


```
aws ec2 describe-instances \  
  --instance-ids i-0123a456700123456 \  
  --query "Reservations[*].Instances[*].Placement"
```

出力:

```
[  
  [  
    {  
      "AvailabilityZone": "us-east-1c",  
      "GroupName": "HDFS-Group-A",  
      "PartitionNumber": 3,  
      "Tenancy": "default"  
    }  
  ]  
]
```

詳細については、「Amazon EC2 ユーザーガイド」の「[プレイスメントグループのインスタンスの説明](#)」を参照してください。

例 12: 指定したプレイスメントグループとパーティション番号を持つインスタンスでフィルタリングするには

次の describe-instances の例では、指定したプレイスメントグループとパーティション番号のインスタンスのみに結果をフィルタリングします。

```
aws ec2 describe-instances \  
  --filters "Name=placement-group-name,Values=HDFS-Group-A" "Name=placement-partition-number,Values=7"
```

次の例では、出力の関連情報のみが表示されます。

```
"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"
        }
    }
],

```

詳細については、「Amazon EC2 ユーザーガイド」の「[プレースメントグループのインスタンスの説明](#)」を参照してください。

例 13: インスタンスメタデータのタグへのアクセスを許可するように設定されているインスタンスに絞り込むには

次の describe-instances の例では、インスタンスメタデータからインスタンスタグへのアクセスを許可するように設定されているインスタンスのみに結果をフィルタリングします。

```

aws ec2 describe-instances \
  --filters "Name=metadata-options.instance-metadata-tags,Values=enabled" \
  --query "Reservations[*].Instances[*].InstanceId" \
  --output text

```

次のような出力が予想されます。

```

i-1234567890abcdefg
i-abcdefg1234567890
i-11111111aaaaaaaaa
i-aaaaaaaaa11111111

```

詳細については、「Amazon EC2 ユーザーガイド」で[インスタンスメタデータ内のインスタンスタグの使用法](#)を参照してください。

- API の詳細については、「AWS CLI Command Reference」の「[DescribeInstances](#)」を参照してください。

Java

SDK for Java 2.x

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/**
 * 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);
    });
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[DescribeInstances](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

/**
 * List all of your EC2 instances running with the provided architecture that
 * were launched in the past month.
 * @param {{ pageSize: string, architectures: string[] }} options
 */
export const main = async ({ pageSize, architectures }) => {
    pageSize = Number.parseInt(pageSize);
    const client = new EC2Client({});
    const d = new Date();
    const year = d.getFullYear();
    const month = `0${d.getMonth() + 1}`.slice(-2);
    const launchTimePattern = `${year}-${month}-*`;

    const paginator = paginateDescribeInstances(
        {
            client,
            pageSize,
        },
        {
            Filters: [
```

```

        { Name: "architecture", Values: architectures },
        { Name: "instance-state-name", Values: ["running"] },
        {
            Name: "launch-time",
            Values: [launchTimePattern],
        },
    ],
},
);

try {
    /**
     * @type {import('@aws-sdk/client-ec2').Instance[]}
     */
    const instanceList = [];
    for await (const page of paginator) {
        const { Reservations } = page;
        for (const reservation of Reservations) {
            instanceList.push(...reservation.Instances);
        }
    }
    console.log(
        `Running instances launched this month:\n\n${instanceList.map((instance) =>
instance.InstanceId).join("\n")}`,
    );
} catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
        console.warn(`${caught.message}.`);
    } else {
        throw caught;
    }
}
};

```

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の「[DescribeInstances](#)」を参照してください。

Kotlin

SDK for Kotlin

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[DescribeInstances](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定したインスタンスを記述します。

```
(Get-EC2Instance -InstanceId i-12345678).Instances
```

出力:

```
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
```

例 2: この例では、現在のリージョンのすべてのインスタンスを予約別にグループ化して記述します。インスタンスの詳細を表示するには、各予約オブジェクト内のインスタンスコレクションを展開します。

```
Get-EC2Instance
```

出力:

```
GroupNames      : {}
Groups          : {}
Instances       : {}
OwnerId         : 123456789012
RequesterId     : 226008221399
ReservationId   : r-c5df370c

GroupNames      : {}
Groups          : {}
Instances       : {}
OwnerId         : 123456789012
RequesterId     : 854251627541
ReservationId   : r-63e65bab
...
```

例 3: この例では、フィルターを使用して VPC の特定のサブネット内の EC2 インスタンスをクエリする方法を示します。

```
(Get-EC2Instance -Filter @{Name="vpc-id";Values="vpc-1a2bc34d"},@{Name="subnet-id";Values="subnet-1a2b3c4d"}).Instances
```

出力:

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		

例 4: この例では、複数の値を持つフィルターを使用して、実行中および停止中の EC2 インスタンスをクエリする方法を示します。

```
$InstanceParams = @{
    Filter = @(
        @{'Name' = 'instance-state-name'; 'Values' = @("running","stopped")}
    )
}

(Get-EC2Instance @InstanceParams).Instances
```

出力:

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		

例 5: この例では、複数の値を持つフィルターを使用して、実行中と停止中の両方の EC2 インスタンスをクエリし、Select-Object コマンドレットを使用して出力する特定の値を選択する方法を示します。

```
$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

```

出力:

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

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeInstances](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定したインスタンスを記述します。

```
(Get-EC2Instance -InstanceId i-12345678).Instances
```

出力:

```

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

```

例 2: この例では、現在のリージョンのすべてのインスタンスを予約別にグループ化して記述します。インスタンスの詳細を表示するには、各予約オブジェクト内のインスタンスコレクションを展開します。

```
Get-EC2Instance
```

出力:

```

GroupNames       : {}
Groups           : {}
Instances        : {}
OwnerId          : 123456789012
RequesterId      : 226008221399
ReservationId    : r-c5df370c

GroupNames       : {}
Groups           : {}

```

```
Instances      : {}
OwnerId       : 123456789012
RequesterId    : 854251627541
ReservationId  : r-63e65bab
...
```

例 3: この例では、フィルターを使用して VPC の特定のサブネット内の EC2 インスタンスをクエリする方法を示します。

```
(Get-EC2Instance -Filter @{{Name="vpc-id";Values="vpc-1a2bc34d"}},{Name="subnet-id";Values="subnet-1a2b3c4d"}}).Instances
```

出力:

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		

例 4: この例では、複数の値を持つフィルターを使用して、実行中および停止中の EC2 インスタンスをクエリする方法を示します。

```
$InstanceParams = @{
    Filter = @(
        @{'Name' = 'instance-state-name'; 'Values' = @("running","stopped")}
    )
}

(Get-EC2Instance @InstanceParams).Instances
```

出力:

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

```

例 5: この例では、複数の値を持つフィルターを使用して、実行中と停止中の両方の EC2 インスタンスをクエリし、Select-Object コマンドレットを使用して出力する特定の値を選択する方法を示します。

```

$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

```

出力:

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

例 6: この例では、実際に EC2 インスタンスを取得することなく、DryRun パラメータを使用して EC2 インスタンス取得のアクセス許可を検証します。注: 成功すると例外がスローされますが、これは正常な動作です。

```
Get-EC2Tag -DryRun $true
```

出力:

```
Get-EC2Instance: Request would have succeeded, but DryRun flag is set.
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeInstances](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
    using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
        instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
        level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
        objects. These are high-level objects that
                           wrap instance actions.
```

```

        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
        EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def display(self, state_filter: Optional[str] = "running") -> None:
        """
        Displays information about instances, filtering by the specified state.

        :param state_filter: The instance state to include in the output. Only
        instances in this state
                               will be displayed. Default is 'running'. Example
        states: 'running', 'stopped'.
        """
        if not self.instances:
            logger.info("No instances to display.")
            return

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        paginator = self.ec2_client.get_paginator("describe_instances")
        page_iterator = paginator.paginate(InstanceIds=instance_ids)

        try:
            for page in page_iterator:
                for reservation in page["Reservations"]:
                    for instance in reservation["Instances"]:
                        instance_state = instance["State"]["Name"]

                        # Apply the state filter (default is 'running')
                        if state_filter and instance_state != state_filter:
                            continue # Skip this instance if it doesn't match
                                the filter

```

```

        # Create a formatted string with instance details
        instance_info = (
            f"• ID: {instance['InstanceId']}\n"
            f"• Image ID: {instance['ImageId']}\n"
            f"• Instance type: {instance['InstanceType']}\n"
            f"• Key name: {instance['KeyName']}\n"
            f"• VPC ID: {instance['VpcId']}\n"
            f"• Public IP: {instance.get('PublicIpAddress', 'N/A')}\n"

            f"• State: {instance_state}"
        )
        print(instance_info)

    except ClientError as err:
        logger.error(
            f"Failed to display instance(s). : {' '.join(map(str, instance_ids))}"
        )
        error_code = err.response["Error"]["Code"]
        if error_code == "InvalidInstanceID.NotFound":
            logger.error(
                "One or more instance IDs do not exist. "
                "Please verify the instance IDs and try again."
            )
            raise

```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[DescribeInstances](#)」を参照してください。

Ruby

SDK for Ruby

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。


```
require 'aws-sdk-ec2'

# @param ec2_resource [Aws::EC2::Resource] An initialized EC2 resource object.
# @example
#   list_instance_ids_states(Aws::EC2::Resource.new(region: 'us-west-2'))
def list_instance_ids_states(ec2_resource)
  response = ec2_resource.instances
  if response.count.zero?
    puts 'No instances found.'
  else
    puts 'Instances -- ID, state:'
    response.each do |instance|
      puts "#{instance.id}, #{instance.state.name}"
    end
  end
end

rescue StandardError => e
  puts "Error getting information about instances: #{e.message}"
end

# Example usage:
def run_me
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-get-all-instance-info.rb REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-get-all-instance-info.rb us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end
  ec2_resource = Aws::EC2::Resource.new(region: region)
  list_instance_ids_states(ec2_resource)
end

run_me if $PROGRAM_NAME == __FILE__
```

- API の詳細については、AWS SDK for Ruby API リファレンスの「[DescribeInstances](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

EC2 インスタンスの詳細を取得します。

```
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())
}
```

EC2 インスタンスを作成した後、その詳細を取得して保存します。

```

/// Create an EC2 instance with the given ID on a given type, using a
/// generated KeyPair and applying a list of security groups.
pub async fn create(
    &mut self,
    ec2: &EC2,
    image_id: &str,
    instance_type: InstanceType,
    key_pair: &KeyPairInfo,
    security_groups: Vec<&SecurityGroup>,
) -> Result<(), EC2Error> {
    let instance_id = ec2
        .create_instance(image_id, instance_type, key_pair, security_groups)
        .await?;
    let instance = ec2.describe_instance(&instance_id).await?;
    self.instance = Some(instance);
    Ok(())
}

```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[DescribeInstances](#)」を参照してください。

SAP ABAP

SDK for SAP ABAP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

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/ec2instancestatename,
           lv_instance_type  TYPE /aws1/ec2instancetype,
           lv_image_id       TYPE /aws1/ec2string.

```

```

        LOOP AT oo_result->get_reservations( ) INTO DATA(lo_reservation).
        LOOP AT lo_reservation->get_instances( ) INTO DATA(lo_instance).
            lv_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.

```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「[DescribeInstances](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeInternetGateways** を使用する

次のサンプルコードは、DescribeInternetGateways を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [Amazon VPC の使用を開始する](#)

CLI

AWS CLI

インターネットゲートウェイを記述するには

次の describe-internet-gateways の例では、指定したインターネットゲートウェイについて説明します。

```
aws ec2 describe-internet-gateways \  
  --internet-gateway-ids igw-0d0fb496b3EXAMPLE
```

出力:

```
{  
  "InternetGateways": [  
    {  
      "Attachments": [  
        {  
          "State": "available",  
          "VpcId": "vpc-0a60eb65b4EXAMPLE"  
        }  
      ],  
      "InternetGatewayId": "igw-0d0fb496b3EXAMPLE",  
      "OwnerId": "123456789012",  
      "Tags": [  
        {  
          "Key": "Name",  
          "Value": "my-igw"  
        }  
      ]  
    }  
  ]  
}
```

詳細については、Amazon VPC ユーザーガイドの「[インターネットゲートウェイ](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeInternetGateways](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたインターネットゲートウェイを記述します。

```
Get-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

出力:

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

例 2: この例では、すべてのインターネットゲートウェイを記述します。

```
Get-EC2InternetGateway
```

出力:

Attachments	InternetGatewayId	Tags
-----	-----	----
{vpc-1a2b3c4d}	igw-1a2b3c4d	{}
{}	igw-2a3b4c5d	{}

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeInternetGateways](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたインターネットゲートウェイを記述します。

```
Get-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

出力:

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

例 2: この例では、すべてのインターネットゲートウェイを記述します。

```
Get-EC2InternetGateway
```

出力:

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

```
{ }          igw-2a3b4c5d          { }
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeInternetGateways](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください [AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DescribeKeyPairs**で を使用する

次のサンプルコードは、DescribeKeyPairs を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

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

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

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

- APIの詳細については、AWS SDK for .NET API リファレンスの「[DescribeKeyPairs](#)」を参照してください。

Bash

AWS CLI Bash スクリプトを使用する

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。


```
#####
# 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
}

```

この例で使用されているユーティリティ関数。

```

#####
# 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
}

```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[DescribeKeyPairs](#)」を参照してください。

C++

SDK for C++

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

//! Describe all Amazon Elastic Compute Cloud (Amazon EC2) instance key pairs.
/*!
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::EC2::describeKeyPairs(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeKeyPairsRequest request;

```

```

    Aws::EC2::Model::DescribeKeyPairsOutcome outcome =
    ec2Client.DescribeKeyPairs(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left <<
            std::setw(32) << "Name" <<
            std::setw(64) << "Fingerprint" << std::endl;

        const std::vector<Aws::EC2::Model::KeyPairInfo> &key_pairs =
            outcome.GetResult().GetKeyPairs();
        for (const auto &key_pair: key_pairs) {
            std::cout << std::left <<
                std::setw(32) << key_pair.GetKeyName() <<
                std::setw(64) << key_pair.GetKeyFingerprint() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe key pairs:" <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- APIの詳細については、AWS SDK for C++ API リファレンスの「[DescribeKeyPairs](#)」を参照してください。

CLI

AWS CLI

キーペアを表示するには

次の describe-key-pairs の例では、指定されたキーペアの情報が表示されます。

```

aws ec2 describe-key-pairs \
    --key-names my-key-pair

```

出力:

```

{
  "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"
    }
]
}

```

詳細については、「Amazon EC2 ユーザーガイド」の「[パブリックキーの説明](#)」を参照してください。

- API の詳細については、「AWS CLI Command Reference」の「[DescribeKeyPairs](#)」を参照してください。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

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

```

```

    }
  });

  return responseFuture;
}

```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[DescribeKeyPairs](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

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

/**
 * List all key pairs in the current AWS account.
 * @param {{ dryRun: boolean }}
 */
export const main = async ({ dryRun }) => {
  const client = new EC2Client({});
  const command = new DescribeKeyPairsCommand({ DryRun: dryRun });

  try {
    const { KeyPairs } = await client.send(command);
    const keyPairList = KeyPairs.map(
      (kp) => ` • ${kp.KeyPairId}: ${kp.KeyName}`,
    ).join("\n");
    console.log("The following key pairs were found in your account:");
    console.log(keyPairList);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "DryRunOperation") {
      console.log(`${caught.message}`);
    } else {

```

```
        throw caught;
    }
}
};
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[DescribeKeyPairs](#)」を参照してください。

Kotlin

SDK for Kotlin

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[DescribeKeyPairs](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定したキーペアを記述します。

```
Get-EC2KeyPair -KeyName my-key-pair
```

出力:

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

例 2: この例では、すべてのキーペアを記述します。

```
Get-EC2KeyPair
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeKeyPairs](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定したキーペアを記述します。

```
Get-EC2KeyPair -KeyName my-key-pair
```

出力:

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

例 2: この例では、すべてのキーペアを記述します。

```
Get-EC2KeyPair
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeKeyPairs](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。


```

class KeyPairWrapper:
    """
    Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions.
    This class provides methods to create, list, and delete EC2 key pairs.
    """

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

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param key_file_dir: The folder where the private key information is
stored.
                           This should be a secure folder.
        :param key_pair: A dictionary representing the Boto3 KeyPair object.
                           This is a high-level object that wraps key pair actions.
Optional.
        """
        self.ec2_client = ec2_client
        self.key_pair = key_pair
        self.key_file_path: Optional[str] = None
        self.key_file_dir = key_file_dir

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

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

```

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

    WARNING: Results are not paginated.

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

        if limit:
            key_pairs = key_pairs[:limit]

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

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[DescribeKeyPairs](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn list_key_pair(&self) -> Result<Vec<KeyPairInfo>, EC2Error> {
    let output = self.client.describe_key_pairs().send().await?;
    Ok(output.key_pairs.unwrap_or_default())
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[DescribeKeyPairs](#)」を参照してください。

SAP ABAP

SDK for SAP ABAP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
TRY.
    oo_result = lo_ec2->describekeypairs( ).
    oo_result is returned for testing purposes. "
    DATA(lt_key_pairs) = oo_result->get_keypairs( ).
    MESSAGE 'Retrieved information about key pairs.' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception->av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「[DescribeKeyPairs](#)」を参照してください。

Swift

SDK for Swift

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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.")
    }
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[DescribeKeyPairs](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeNetworkAcls** を使用する

次のサンプルコードは、DescribeNetworkAcls を使用する方法を説明しています。

CLI

AWS CLI

ネットワーク ACL を記述するには

次の describe-network-acls の例では、ネットワーク ACL の詳細を取得します。

```
aws ec2 describe-network-acls
```

出力:

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

詳細については、「AWS VPC ユーザーガイド」の「[Network ACLs](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeNetworkAcls](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定したネットワーク ACL を記述します。

```
Get-EC2NetworkAcl -NetworkAclId acl-12345678
```

出力:

```

Associations : {aclassoc-1a2b3c4d}
Entries       : {Amazon.EC2.Model.NetworkAclEntry,
  Amazon.EC2.Model.NetworkAclEntry}
IsDefault     : False
NetworkAclId  : acl-12345678
Tags          : {Name}
VpcId         : vpc-12345678
```

例 2: この例では、指定されたネットワーク ACL のルールを記述します。

```
(Get-EC2NetworkAcl -NetworkAclId acl-12345678).Entries
```

出力:

```

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

```

例 3: この例では、すべてのネットワーク ACL を記述します。

```
Get-EC2NetworkAcl
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeNetworkAcls](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定したネットワーク ACL を記述します。

```
Get-EC2NetworkAcl -NetworkAclId acl-12345678
```

出力:

```

Associations : {aclassoc-1a2b3c4d}
Entries      : {Amazon.EC2.Model.NetworkAclEntry,
  Amazon.EC2.Model.NetworkAclEntry}
IsDefault    : False
NetworkAclId : acl-12345678
Tags         : {Name}
VpcId        : vpc-12345678

```

例 2: この例では、指定されたネットワーク ACL のルールを記述します。

```
(Get-EC2NetworkAcl -NetworkAclId acl-12345678).Entries
```

出力:

```
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
```

例 3: この例では、すべてのネットワーク ACL を記述します。

```
Get-EC2NetworkAcl
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeNetworkAcls](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください [AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeNetworkInterfaceAttribute** を使用する

次のサンプルコードは、**DescribeNetworkInterfaceAttribute** を使用する方法を説明しています。

CLI

AWS CLI

ネットワークインターフェイスのアタッチメント属性を記述するには

このコマンド例では、指定されたネットワークインターフェイスの `attachment` 属性を記述します。

コマンド:

```
aws ec2 describe-network-interface-attribute --network-interface-id eni-686ea200
--attribute attachment
```

出力:

```
{
  "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"
  }
}
```

ネットワークインターフェイスの説明属性を記述するには

このコマンド例では、指定されたネットワークインターフェイスの `description` 属性を記述します。

コマンド:

```
aws ec2 describe-network-interface-attribute --network-interface-id eni-686ea200
--attribute description
```

出力:

```
{
  "NetworkInterfaceId": "eni-686ea200",
  "Description": {
    "Value": "My description"
  }
}
```

ネットワークインターフェイスの `groupSet` 属性を記述するには

このコマンド例では、指定されたネットワークインターフェイスの `groupSet` 属性を記述します。

コマンド:

```
aws ec2 describe-network-interface-attribute --network-interface-id eni-686ea200
--attribute groupSet
```

出力:

```
{
  "NetworkInterfaceId": "eni-686ea200",
  "Groups": [
    {
      "GroupName": "my-security-group",
      "GroupId": "sg-903004f8"
    }
  ]
}
```

ネットワークインターフェイスの `sourceDestCheck` 属性を記述するには

このコマンド例では、指定されたネットワークインターフェイスの `sourceDestCheck` 属性を記述します。

コマンド:

```
aws ec2 describe-network-interface-attribute --network-interface-id eni-686ea200
--attribute sourceDestCheck
```

出力:

```
{
  "NetworkInterfaceId": "eni-686ea200",
  "SourceDestCheck": {
    "Value": true
  }
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeNetworkInterfaceAttribute](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたネットワークインターフェイスを記述します。

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute Attachment
```

出力:

```
Attachment           : Amazon.EC2.Model.NetworkInterfaceAttachment
```

例 2: この例では、指定されたネットワークインターフェイスを記述します。

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute Description
```

出力:

```
Description          : My description
```

例 3: この例では、指定されたネットワークインターフェイスを記述します。

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute GroupSet
```

出力:

```
Groups               : {my-security-group}
```

例 4: この例では、指定されたネットワークインターフェイスを記述します。

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute SourceDestCheck
```

出力:

```
SourceDestCheck      : True
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeNetworkInterfaceAttribute](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたネットワークインターフェイスを記述します。

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute Attachment
```

出力:

```
Attachment          : Amazon.EC2.Model.NetworkInterfaceAttachment
```

例 2: この例では、指定されたネットワークインターフェイスを記述します。

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute Description
```

出力:

```
Description         : My description
```

例 3: この例では、指定されたネットワークインターフェイスを記述します。

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute GroupSet
```

出力:

```
Groups              : {my-security-group}
```

例 4: この例では、指定されたネットワークインターフェイスを記述します。

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute SourceDestCheck
```

出力:

```
SourceDestCheck     : True
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeNetworkInterfaceAttribute](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeNetworkInterfaces** を使用する

次のサンプルコードは、DescribeNetworkInterfaces を使用する方法を説明しています。

CLI

AWS CLI

ネットワークインターフェイスを記述するには

この例では、すべてのネットワークインターフェイスを記述します。

コマンド:

```
aws ec2 describe-network-interfaces
```

出力:

```
{
  "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"
    }
]
}

```

この例では、キー `Purpose` と値 `Prod` を含むタグを持つネットワークインターフェイスを記述します。

コマンド:

```
aws ec2 describe-network-interfaces --filters Name=tag:Purpose,Values=Prod
```

出力:

```
{
  "NetworkInterfaces": [
    {
      "Status": "available",
      "MacAddress": "12:2c:bd:f9:bf:17",
      "SourceDestCheck": true,
      "VpcId": "vpc-8941ebec",
      "Description": "ProdENI",
      "NetworkInterfaceId": "eni-b9a5ac93",
      "PrivateIpAddresses": [
        {
          "PrivateDnsName": "ip-10-0-1-55.ec2.internal",
          "Primary": true,
          "PrivateIpAddress": "10.0.1.55"
        },
        {
          "PrivateDnsName": "ip-10-0-1-117.ec2.internal",
          "Primary": false,
          "PrivateIpAddress": "10.0.1.117"
        }
      ],
      "RequesterManaged": false,
      "PrivateDnsName": "ip-10-0-1-55.ec2.internal",
      "AvailabilityZone": "us-east-1d",
      "Ipv6Addresses": [],
      "Groups": [
        {
          "GroupName": "MySG",
          "GroupId": "sg-905002f5"
        }
      ],
      "SubnetId": "subnet-31d6c219",
      "OwnerId": "123456789012",
      "TagSet": [
        {
          "Value": "Prod",
          "Key": "Purpose"
        }
      ],
    }
  ],
}
```

```
        "PrivateIpAddress": "10.0.1.55"
    }
]
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeNetworkInterfaces](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたネットワークインターフェイスを記述します。

```
Get-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

出力:

```
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
```

例 2: この例では、すべてのネットワークインターフェイスを記述します。

```
Get-EC2NetworkInterface
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeNetworkInterfaces](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたネットワークインターフェイスを記述します。

```
Get-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

出力:

```
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
```

例 2: この例では、すべてのネットワークインターフェイスを記述します。

```
Get-EC2NetworkInterface
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeNetworkInterfaces](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribePlacementGroups** を使用する

次のサンプルコードは、DescribePlacementGroups を使用する方法を説明しています。

CLI

AWS CLI

プレイズメントグループを記述するには

このコマンド例では、すべてのプレイズメントグループを記述します。

コマンド:

```
aws ec2 describe-placement-groups
```

出力:

```
{
  "PlacementGroups": [
    {
      "GroupName": "my-cluster",
      "State": "available",
      "Strategy": "cluster"
    },
    ...
  ]
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribePlacementGroups](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定したプレイズメントグループを記述します。

```
Get-EC2PlacementGroup -GroupName my-placement-group
```

出力:

GroupName	State	Strategy
-----	-----	-----
my-placement-group	available	cluster

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribePlacementGroups](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定したプレースメントグループを記述します。

```
Get-EC2PlacementGroup -GroupName my-placement-group
```

出力:

GroupName	State	Strategy
-----	-----	-----
my-placement-group	available	cluster

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribePlacementGroups](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribePrefixLists** を使用する

次のサンプルコードは、DescribePrefixLists を使用方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [プライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成する](#)

CLI

AWS CLI

プレフィックスリストを記述するには

この例では、リージョンで使用可能なすべてのプレフィックスリストを一覧表示します。

コマンド:

```
aws ec2 describe-prefix-lists
```

出力:

```
{
  "PrefixLists": [
    {
      "PrefixListName": "com.amazonaws.us-east-1.s3",
      "Cidrs": [
        "54.231.0.0/17"
      ],
      "PrefixListId": "pl-63a5400a"
    }
  ]
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribePrefixLists](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、リージョン AWS のサービスのプレフィックスリスト形式で使用可能なを取得します。

```
Get-EC2PrefixList
```

出力:

Cidrs	PrefixListId	PrefixListName
-----	-----	-----
{52.94.5.0/24, 52.119.240.0/21, 52.94.24.0/23}	pl-6fa54006	com.amazonaws.eu-west-1.dynamodb
{52.218.0.0/17, 54.231.128.0/19}	pl-6da54004	com.amazonaws.eu-west-1.s3

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribePrefixLists](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、リージョン AWS のサービスのプレフィックスリスト形式で使用可能なを取得します。

```
Get-EC2PrefixList
```

出力:

Cidrs	PrefixListId	PrefixListName
-----	-----	-----
{52.94.5.0/24, 52.119.240.0/21, 52.94.24.0/23}	pl-6fa54006	com.amazonaws.eu-west-1.dynamodb
{52.218.0.0/17, 54.231.128.0/19}	pl-6da54004	com.amazonaws.eu-west-1.s3

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribePrefixLists](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DescribeRegions**で を使用する

次のサンプルコードは、DescribeRegions を使用する方法を説明しています。

C++

SDK for C++

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
//! Describe all Amazon Elastic Compute Cloud (Amazon EC2) Regions.
```



```

/ * !
\ param clientConfiguration: AWS client configuration.
\ return bool: Function succeeded.
*/
bool AwsDoc::EC2::describeRegions(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::DescribeRegionsRequest request;
    Aws::EC2::Model::DescribeRegionsOutcome outcome =
ec2Client.DescribeRegions(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left <<
            std::setw(32) << "RegionName" <<
            std::setw(64) << "Endpoint" << std::endl;

        const auto &regions = outcome.GetResult().GetRegions();
        for (const auto &region: regions) {
            std::cout << std::left <<
                std::setw(32) << region.GetRegionName() <<
                std::setw(64) << region.GetEndpoint() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe regions:" <<
            outcome.GetError().GetMessage() << std::endl;
    }

    std::cout << std::endl;

    return outcome.IsSuccess();
}

```

- APIの詳細については、AWS SDK for C++ API リファレンスの「[DescribeRegions](#)」を参照してください。

CLI

AWS CLI

例 1: 有効になっているすべてのリージョンを説明するには

次の `describe-regions` の例は、アカウントで有効なすべてのリージョンを説明しています。

```
aws ec2 describe-regions
```

出力:

```
{
  "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"
    }
]
}

```

詳細については、「Amazon EC2 ユーザーガイド」の「[リージョンとゾーン](#)」を参照してください。

例 2: エンドポイント名に特定の文字列が含まれる有効なリージョンを説明するには

次の describe-regions の例では、エンドポイントに「us」という文字列が含まれる、有効にしたすべてのリージョンを説明しています。

```

aws ec2 describe-regions \
  --filters "Name=endpoint,Values=*us*"

```

出力:

```

{
  "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"
    }
  ]
}

```

```
}
```

詳細については、「Amazon EC2 ユーザーガイド」の「[リージョンとゾーン](#)」を参照してください。

例 3: すべてのリージョンを説明するには

次の describe-regions の例では、無効になっているリージョンを含め、使用可能なすべてのリージョンについて説明しています。

```
aws ec2 describe-regions \  
  --all-regions
```

出力:

```
{  
  "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"
}
]
```

詳細については、「Amazon EC2 ユーザーガイド」の「[リージョンとゾーン](#)」を参照してください。

例 4: リージョン名だけを一覧表示するには

次の describe-regions の例では、--query パラメータを使用して出力をフィルタリングし、リージョンの名前のみをテキストとして返します。

```
aws ec2 describe-regions \
  --all-regions \
  --query "Regions[].{Name:RegionName}" \
  --output text
```

出力:

```
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
```

詳細については、「Amazon EC2 ユーザーガイド」の「[リージョンとゾーン](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeRegions](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

/**
 * List all available AWS regions.
 * @param {{ regionNames: string[], includeOptInRegions: boolean }} options
```



```
*/
export const main = async ({ regionNames, includeOptInRegions }) => {
  const client = new EC2Client({});
  const command = new DescribeRegionsCommand({
    // By default this command will not show regions that require you to opt-in.
    // When AllRegions is true, even the regions that require opt-in will be
    returned.
    AllRegions: includeOptInRegions,
    // You can omit the Filters property if you want to get all regions.
    Filters: regionNames?.length
      ? [
          {
            Name: "region-name",
            // You can specify multiple values for a filter.
            // You can also use '*' as a wildcard. This will return all
            // of the regions that start with 'us-east-'.
            Values: regionNames,
          },
        ]
      : undefined,
  });

  try {
    const { Regions } = await client.send(command);
    const regionsList = Regions.map((reg) => ` • ${reg.RegionName}`);
    console.log("Found regions:");
    console.log(regionsList.join("\n"));
  } catch (caught) {
    if (caught instanceof Error && caught.name === "DryRunOperation") {
      console.log(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- API の詳細については、AWS SDK for JavaScript API リファレンスの「[DescribeRegions](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、使用可能なリージョンについて記述します。

```
Get-EC2Region
```

出力:

Endpoint	RegionName
-----	-----
ec2.eu-west-1.amazonaws.com	eu-west-1
ec2.ap-southeast-1.amazonaws.com	ap-southeast-1
ec2.ap-southeast-2.amazonaws.com	ap-southeast-2
ec2.eu-central-1.amazonaws.com	eu-central-1
ec2.ap-northeast-1.amazonaws.com	ap-northeast-1
ec2.us-east-1.amazonaws.com	us-east-1
ec2.sa-east-1.amazonaws.com	sa-east-1
ec2.us-west-1.amazonaws.com	us-west-1
ec2.us-west-2.amazonaws.com	us-west-2

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeRegions](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、使用可能なリージョンについて記述します。

```
Get-EC2Region
```

出力:

Endpoint	RegionName
-----	-----
ec2.eu-west-1.amazonaws.com	eu-west-1
ec2.ap-southeast-1.amazonaws.com	ap-southeast-1
ec2.ap-southeast-2.amazonaws.com	ap-southeast-2
ec2.eu-central-1.amazonaws.com	eu-central-1
ec2.ap-northeast-1.amazonaws.com	ap-northeast-1
ec2.us-east-1.amazonaws.com	us-east-1
ec2.sa-east-1.amazonaws.com	sa-east-1
ec2.us-west-1.amazonaws.com	us-west-1

ec2.us-west-2.amazonaws.com

us-west-2

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeRegions](#)」を参照してください。

Ruby

SDK for Ruby

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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
```

```

    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
  end
end

```

```
        end
        print "\n"
    end
end

# Example usage:
def run_me
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:  ruby ec2-ruby-example-regions-availability-zones.rb REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-regions-availability-zones.rb us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts 'AWS Regions for Amazon EC2 that are available to you:'
  list_regions_endpoints(ec2_client)
  puts "\n\nAmazon EC2 Availability Zones that are available to you for AWS
  Region '#{region}':"
  list_availability_zones(ec2_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[DescribeRegions](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
async fn show_regions(client: &Client) -> Result<(), Error> {
    let rsp = client.describe_regions().send().await?;

    println!("Regions:");
    for region in rsp.regions() {
        println!("  {}", region.region_name().unwrap());
    }

    Ok(())
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[DescribeRegions](#)」を参照してください。

SAP ABAP

SDK for SAP ABAP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
TRY.
    oo_result = lo_ec2->describeregions( ).
    "
    oo_result is returned for testing purposes. "
    DATA(lt_regions) = oo_result->get_regions( ).
    MESSAGE 'Retrieved information about Regions.' TYPE 'I'.
```

```
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
  DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
  MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「[DescribeRegions](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DescribeRouteTables**で を使用する

次のサンプルコードは、DescribeRouteTables を使用方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [Amazon VPC の使用を開始する](#)
- [Transit Gateway を開始する](#)

CLI

AWS CLI

ルートテーブルを記述するには

次の describe-route-tables の例では、ルートテーブルの詳細を取得します。

```
aws ec2 describe-route-tables
```

出力:

```
{
  "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"
}
]
}

```

詳細については、「AWS VPC ユーザーガイド」の「[Working with Route Tables](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeRouteTables](#)」を参照してください。

PHP

SDK for PHP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/**
 * @param array $routeTableIds
 * @param array $filters
 * @return array
 */
public function describeRouteTables(array $routeTableIds = [], array $filters
= []): array
{
    $parameters = [];
    if($routeTableIds){
        $parameters['RouteTableIds'] = $routeTableIds;
    }
    if($filters){
        $parameters['Filters'] = $filters;
    }
    try {
        $paginator = $this->ec2Client->getPaginator("DescribeRouteTables",
$parameters);
        $contents = [];
        foreach ($paginator as $result) {
            foreach ($result['RouteTables'] as $object) {
                $contents[] = $object['RouteTableId'];
            }
        }
    } catch (Ec2Exception $caught){
```

```

        echo "There was a problem paginating the results of
DescribeRouteTables: {$caught->getAwsErrorMessage()}\n";
        throw $caught;
    }
    return $contents;
}

```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「[DescribeRouteTables](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、すべてのルートテーブルについて記述します。

```
Get-EC2RouteTable
```

出力:

```

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 :

```

例 2: この例では、指定されたルートテーブルの詳細を返します。

```
Get-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

例 3: この例では、指定された VPC のルートテーブルを記述します。

```
Get-EC2RouteTable -Filter @{ Name="vpc-id"; Values="vpc-1a2b3c4d" }
```

出力:

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeRouteTables](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、すべてのルートテーブルについて記述します。

```
Get-EC2RouteTable
```

出力:

```
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 :

```

例 2: この例では、指定されたルートテーブルの詳細を返します。

```
Get-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

例 3: この例では、指定された VPC のルートテーブルを記述します。

```
Get-EC2RouteTable -Filter @{ Name="vpc-id"; Values="vpc-1a2b3c4d" }
```

出力:

```

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

```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeRouteTables](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

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

```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[DescribeRouteTables](#)」を参照してください。

SAP ABAP

SDK for SAP ABAP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
" 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.
```

- API の詳細については、AWS SDK for SAP ABAP API リファレンスの「[DescribeRouteTables](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で `DescribeScheduledInstanceAvailability` を使用する

次のサンプルコードは、`DescribeScheduledInstanceAvailability` を使用する方法を説明しています。

CLI

AWS CLI

使用可能なスケジュールを記述するには

この例では、指定した日付から毎週日曜日に発生するスケジュールを記述します。

コマンド:

```
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
```

出力:

```
{  
  "ScheduledInstanceAvailabilitySet": [  
    {  
      "AvailabilityZone": "us-west-2b",  
      "TotalScheduledInstanceHours": 1219,  
      "PurchaseToken": "eyJ2IjoiMSIsInMiOiEsImMiOi...",  
      "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"
    },
    ...
]
}

```

結果を絞り込むために、オペレーティングシステム、ネットワーク、インスタンスタイプを指定するフィルターを追加できます。

コマンド:

```
--filters Name=platform,Values=Linux/UNIX Name=network-platform,Values=EC2-VPC
Name=instance-type,Values=c4.large
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeScheduledInstanceAvailability](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定した日付から毎週日曜日に発生するスケジュールを記述します。

```

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

```

出力:

```

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
...
```

例 2: 結果を絞り込むには、オペレーティングシステム、ネットワーク、インスタンスタイプなどの条件でフィルターを追加できます。

```
-Filter @{ Name="platform";Values="Linux/UNIX" },@{ Name="network-
platform";Values="EC2-VPC" },@{ Name="instance-type";Values="c4.large" }
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeScheduledInstanceAvailability](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定した日付から毎週日曜日に発生するスケジュールを記述します。

```
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
```

出力:

```
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
...
```

例 2: 結果を絞り込むには、オペレーティングシステム、ネットワーク、インスタンスタイプなどの条件でフィルターを追加できます。

```
-Filter @{ Name="platform";Values="Linux/UNIX" },@{ Name="network-  
platform";Values="EC2-VPC" },@{ Name="instance-type";Values="c4.large" }
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeScheduledInstanceAvailability](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeScheduledInstances** を使用する

次のサンプルコードは、DescribeScheduledInstances を使用する方法を説明しています。

CLI

AWS CLI

スケジュールされたインスタンスを記述するには

この例では、指定のスケジュールされたインスタンスを記述します。

コマンド:

```
aws ec2 describe-scheduled-instances --scheduled-instance-  
ids sci-1234-1234-1234-123456789012
```

出力:

```
{  
  "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"
    }
]
}

```

この例では、すべてのスケジュールされたインスタンスを記述します。

コマンド:

```
aws ec2 describe-scheduled-instances
```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[DescribeScheduledInstances](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定のスケジュールされたインスタンスを記述します。

```
Get-EC2ScheduledInstance -ScheduledInstanceId
sci-1234-1234-1234-1234-123456789012
```

出力:

```

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

```

例 2: この例では、すべてのスケジュールされたインスタンスを記述します。

```
Get-EC2ScheduledInstance
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeScheduledInstances](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定のスケジュールされたインスタンスを記述します。

```
Get-EC2ScheduledInstance -ScheduledInstanceId
sci-1234-1234-1234-123456789012
```

出力:

```

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
```

例 2: この例では、すべてのスケジュールされたインスタンスを記述します。

```
Get-EC2ScheduledInstance
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeScheduledInstances](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DescribeSecurityGroups**で を使用する

次のサンプルコードは、DescribeSecurityGroups を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)
- [Amazon VPC の使用を開始する](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

- API の詳細については、AWS SDK for .NET API リファレンスの「[DescribeSecurityGroups](#)」を参照してください。

Bash

AWS CLI Bash スクリプトを使用する

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

この例で使用されているユーティリティ関数。

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

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[DescribeSecurityGroups](#)」を参照してください。

C++

SDK for C++

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
//! Describe all Amazon Elastic Compute Cloud (Amazon EC2) security groups, or a
specific group.
/*!
    \param groupID: A group ID, ignored if empty.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::EC2::describeSecurityGroups(const Aws::String &groupID,
                                         const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeSecurityGroupsRequest request;

    if (!groupID.empty()) {
        request.AddGroupIds(groupID);
    }

    Aws::String nextToken;
    do {
        if (!nextToken.empty()) {
```

```
        request.SetNextToken(nextToken);
    }

    Aws::EC2::Model::DescribeSecurityGroupsOutcome outcome =
    ec2Client.DescribeSecurityGroups(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left <<
            std::setw(32) << "Name" <<
            std::setw(30) << "GroupId" <<
            std::setw(30) << "VpcId" <<
            std::setw(64) << "Description" << std::endl;

        const std::vector<Aws::EC2::Model::SecurityGroup> &securityGroups =
            outcome.GetResult().GetSecurityGroups();

        for (const auto &securityGroup: securityGroups) {
            std::cout << std::left <<
                std::setw(32) << securityGroup.GetGroupName() <<
                std::setw(30) << securityGroup.GetGroupId() <<
                std::setw(30) << securityGroup.GetVpcId() <<
                std::setw(64) << securityGroup.GetDescription() <<
                std::endl;
        }
    } else {
        std::cerr << "Failed to describe security groups:" <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    nextToken = outcome.GetResult().GetNextToken();
} while (!nextToken.empty());

return true;
}
```

- APIの詳細については、AWS SDK for C++ API リファレンスの「[DescribeSecurityGroups](#)」を参照してください。

CLI

AWS CLI

例 1: セキュリティグループを説明するには

次の `describe-security-groups` の例では、指定したセキュリティグループを示しています。

```
aws ec2 describe-security-groups \
  --group-ids sg-903004f8
```

出力:

```
{
  "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",
}
]
}

```

例 2: 特定のルールを持つセキュリティグループを説明するには

次の `describe-security-groups` の例では、フィルターを使用して、SSH トラフィック (ポート 22) を許可するルールと、すべてのアドレス (`0.0.0.0/0`) からのトラフィックを許可するルールを持つセキュリティグループに結果の範囲を絞り込みます。例では、`--query` パラメータを使用してセキュリティグループの名前のみを表示しています。セキュリティグループが結果で返されるようにするには、すべてのフィルターに一致する必要があります。ただし、1 つのルールがすべてのフィルターに一致する必要はありません。例えば、出力は、特定の IP アドレスからの SSH トラフィックを許可するルールと、すべてのアドレスからの HTTP トラフィックを許可する別のルールを含むセキュリティグループを返します。

```

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
```

出力:

```
default
my-security-group
web-servers
launch-wizard-1
```

例 3: タグに基づいてセキュリティグループを説明するには

次の describe-security-groups の例では、フィルターを使用して、セキュリティグループ名に test が含まれ、タグ Test=To-delete が付けられているセキュリティグループに結果を絞り込みます。例では、--query パラメータを使用してセキュリティグループの名前と ID のみを表示しています。

```
aws ec2 describe-security-groups \
  --filters Name=group-name,Values=*test* Name=tag:Test,Values=To-delete \
  --query "SecurityGroups[*].{Name:GroupName,ID:GroupId}"
```

出力:

```
[
  {
    "Name": "testfornewinstance",
    "ID": "sg-33bb22aa"
  },
  {
    "Name": "newgrouptest",
    "ID": "sg-1a2b3c4d"
  }
]
```

タグフィルターを使用するその他の例については、「Amazon EC2 ユーザーガイド」の「[タグの使用](#)」を参照してください。

- API の詳細については、「AWS CLI Command Reference」の「[DescribeSecurityGroups](#)」を参照してください。

Java

SDK for Java 2.x

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

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

```

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

```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[DescribeSecurityGroups](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

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

/**
 * Describes the specified security groups or all of your security groups.
 * @param {{ groupId: string[] }} options
 */
export const main = async ({ groupId = [] }) => {
    const client = new EC2Client({});
    const command = new DescribeSecurityGroupsCommand({
        GroupIds: groupId,
    });

    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;
    }
}
};

```

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の「[DescribeSecurityGroups](#)」を参照してください。

Kotlin

SDK for Kotlin

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

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}")
}
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[DescribeSecurityGroups](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、VPC に指定されたセキュリティグループについて記述します。VPC に属するセキュリティグループを使用する場合は、グループ名 (-GroupName パラメータ) ではなくセキュリティグループ ID (-GroupId パラメータ) を使用してグループを参照する必要があります。

```
Get-EC2SecurityGroup -GroupId sg-12345678
```

出力:

```
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
```

例 2: この例では、EC2-Classic に指定されたセキュリティグループを記述します。EC2-Classic のセキュリティグループを使用する場合は、グループ名 (-GroupName パラメータ) またはグループ ID (-GroupId パラメータ) を使用してセキュリティグループを参照できます。

```
Get-EC2SecurityGroup -GroupName my-security-group
```

出力:

```

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           :

```

例 3: この例では、vpc-0fc1ff23456b789eb のすべてのセキュリティグループを取得します。

```
Get-EC2SecurityGroup -Filter @{Name="vpc-id";Values="vpc-0fc1ff23456b789eb"}
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeSecurityGroups](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、VPC に指定されたセキュリティグループについて記述します。VPC に属するセキュリティグループを使用する場合は、グループ名 (-GroupName パラメータ) ではなくセキュリティグループ ID (-GroupId パラメータ) を使用してグループを参照する必要があります。

```
Get-EC2SecurityGroup -GroupId sg-12345678
```

出力:

```

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

```

例 2: この例では、EC2-Classic に指定されたセキュリティグループを記述します。EC2-Classic のセキュリティグループを使用する場合は、グループ名 (-GroupName パラメータ) またはグループ ID (-GroupId パラメータ) を使用してセキュリティグループを参照できます。

```
Get-EC2SecurityGroup -GroupName my-security-group
```

出力:

```
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           :
```

例 3: この例では、vpc-0fc1ff23456b789eb のすべてのセキュリティグループを取得します。

```
Get-EC2SecurityGroup -Filter @{"Name"="vpc-id";Values="vpc-0fc1ff23456b789eb"}
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeSecurityGroups](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
    None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
        security group ID.
```

```

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                                access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a
high-level identifier
                                that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.

        :return: An instance of SecurityGroupWrapper initialized with the default
EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def describe(self, security_group_id: Optional[str] = None) -> bool:
        """
        Displays information about the specified security group or all security
groups if no ID is provided.

        :param security_group_id: The ID of the security group to describe.
                                If None, an open search is performed to
describe all security groups.
        :returns: True if the description is successful.
        :raises ClientError: If there is an error describing the security
group(s), such as an invalid security group ID.
        """
        try:
            paginator = self.ec2_client.get_paginator("describe_security_groups")

            if security_group_id is None:
                # If no ID is provided, return all security groups.
                page_iterator = paginator.paginate()
            else:
                page_iterator = paginator.paginate(GroupIds=[security_group_id])

```

```

    for page in page_iterator:
        for security_group in page["SecurityGroups"]:
            print(f"Security group: {security_group['GroupName']}")
            print(f"\tID: {security_group['GroupId']}")
            print(f"\tVPC: {security_group['VpcId']}")
            if security_group["IpPermissions"]:
                print("Inbound permissions:")
                pp(security_group["IpPermissions"])

    return True
except ClientError as err:
    logger.error("Failed to describe security group(s).")
    if err.response["Error"]["Code"] == "InvalidGroup.NotFound":
        logger.error(
            f"Security group {security_group_id} does not exist "
            f"because the specified security group ID was not found."
        )
    raise

```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[DescribeSecurityGroups](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

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

```



```

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

```

- APIの詳細については、「AWS SDK for Rust API リファレンス」の「[DescribeSecurityGroups](#)」を参照してください。

SAP ABAP

SDK for SAP ABAP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

TRY.

```

DATA lt_group_ids TYPE /aws1/
cl_ec2groupidstrlist_w=>tt_groupidstringlist.

```

```

        APPEND NEW /aws1/cl_ec2groupidstrlist_w( iv_value = iv_group_id ) TO
lt_group_ids.
        oo_result = lo_ec2->describesecuritygroups( it_groupids = lt_group_ids ).
        " oo_result is returned for testing purposes. "
        DATA(lt_security_groups) = oo_result->get_securitygroups( ).
        MESSAGE 'Retrieved information about security groups.' TYPE 'I'.
        CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.

```

- APIの詳細については、「AWS SDK for SAP ABAP API リファレンス」の「[DescribeSecurityGroups](#)」を参照してください。

Swift

SDK for Swift

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

`describeSecurityGroupsPaginated()` を使ったページ分割の使用。

```

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
}
```

ページ分割なし。

```
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
    }
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[DescribeSecurityGroups](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeSnapshotAttribute** を使用する

次のサンプルコードは、DescribeSnapshotAttribute を使用する方法を説明しています。

CLI

AWS CLI

スナップショットのスナップショット属性を記述するには

次の describe-snapshot-attribute の例では、スナップショットを共有するアカウントを一覧表示します。

```
aws ec2 describe-snapshot-attribute \
  --snapshot-id snap-01234567890abcdef \
  --attribute createVolumePermission
```

出力:

```
{
  "SnapshotId": "snap-01234567890abcdef",
  "CreateVolumePermissions": [
    {
      "UserId": "123456789012"
    }
  ]
}
```

```
]
}
```

詳細については、「Amazon Elastic Compute Cloud ユーザーガイド」の「[Share an Amazon EBS snapshot](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeSnapshotAttribute](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたスナップショットの指定された属性を記述します。

```
Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute ProductCodes
```

出力:

CreateVolumePermissions	ProductCodes	SnapshotId
-----	-----	-----
{}	{}	snap-12345678

例 2: この例では、指定されたスナップショットの指定された属性を記述します。

```
(Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
CreateVolumePermission).CreateVolumePermissions
```

出力:

Group	UserId
-----	-----
all	

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeSnapshotAttribute](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたスナップショットの指定された属性を記述します。

```
Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute ProductCodes
```

出力:

CreateVolumePermissions	ProductCodes	SnapshotId
-----	-----	-----
{}	{}	snap-12345678

例 2: この例では、指定されたスナップショットの指定された属性を記述します。

```
(Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
CreateVolumePermission).CreateVolumePermissions
```

出力:

Group	UserId
-----	-----
all	

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeSnapshotAttribute](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DescribeSnapshots**で を使用する

次のサンプルコードは、DescribeSnapshots を使用する方法を説明しています。

CLI

AWS CLI

例 1: スナップショットを説明するには

次の describe-snapshots の例では、指定したスナップショットを示しています。

```
aws ec2 describe-snapshots \
  --snapshot-ids snap-1234567890abcdef0
```

出力:

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

詳細については、「Amazon EC2 ユーザーガイド」の「[Amazon EBS スナップショット](#)」を参照してください。

例 2: フィルターに基づいてスナップショットを説明するには

次のdescribe-snapshots例では、フィルターを使用して、pending状態にある AWS アカウントが所有するスナップショットに結果をスコープします。この例では、--query パラメータを使用して、スナップショット ID とスナップショットが開始された時間のみを表示します。

```
aws ec2 describe-snapshots \
  --owner-ids self \
  --filters Name=status,Values=pending \
  --query "Snapshots[*].{ID:SnapshotId,Time:StartTime}"
```

出力:

```
[
```

```
{
  "ID": "snap-1234567890abcdef0",
  "Time": "2019-08-04T12:48:18.000Z"
},
{
  "ID": "snap-066877671789bd71b",
  "Time": "2019-08-04T02:45:16.000Z"
},
...
]
```

次の describe-snapshots の例では、フィルターを使用して、指定したボリュームから作成されたスナップショットに結果を絞っています。この例では、--query パラメータを使用してスナップショット ID のみを表示します。

```
aws ec2 describe-snapshots \
  --filters Name=volume-id,Values=049df61146c4d7901 \
  --query "Snapshots[*].[SnapshotId]" \
  --output text
```

出力:

```
snap-1234567890abcdef0
snap-08637175a712c3fb9
...
```

フィルターを使用するその他の例については、「Amazon EC2 ユーザーガイド」の「[リソースの一覧表示とフィルタリングの方法](#)」を参照してください。

例 3: タグに基づいてスナップショットを説明するには

次の describe-snapshots の例では、タグフィルターを使用して、結果の範囲をタグ Stack=Prod を含むスナップショットに限定しています。

```
aws ec2 describe-snapshots \
  --filters Name=tag:Stack,Values=prod
```

describe-snapshots の出力例については、例 1 を参照してください。

タグフィルターを使用するその他の例については、「Amazon EC2 ユーザーガイド」の「[タグの使用法](#)」を参照してください。

例 4: 日付に基づいてスナップショットを説明するには

次のdescribe-snapshots例では、JMESPath 式を使用して、指定した日付より前に AWS アカウントによって作成されたすべてのスナップショットを記述します。スナップショット ID のみが表示されます。

```
aws ec2 describe-snapshots \  
  --owner-ids 012345678910 \  
  --query "Snapshots[?(StartTime<='2020-03-31')].[SnapshotId]"
```

フィルターを使用するその他の例については、「Amazon EC2 ユーザーガイド」の「[リソースの一覧表示とフィルタリングの方法](#)」を参照してください。

例 5: アーカイブされたスナップショットのみを表示するには

次の describe-snapshots の例では、アーカイブ階層に保存されたスナップショットのみを説明しています。

```
aws ec2 describe-snapshots \  
  --filters "Name=storage-tier,Values=archive"
```

出力:

```
{  
  "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": []  
    },  
  ]  
}
```

詳細については、「Amazon Elastic Compute Cloud ユーザーガイド」の「[アーカイブされたスナップショットを表示する](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeSnapshots](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定したスナップショットを記述します。

```
Get-EC2Snapshot -SnapshotId snap-12345678
```

出力:

```
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
```

例 2: この例では、「Name」タグを持つスナップショットを記述します。

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" }
```

例 3: この例では、「Name」タグと値「TestValue」を持つスナップショットを記述します。

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" -and
                        $_.Tags.Value -eq "TestValue" }
```

例 4: この例では、すべてのスナップショットを記述します。

```
Get-EC2Snapshot -Owner self
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeSnapshots](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定したスナップショットを記述します。

```
Get-EC2Snapshot -SnapshotId snap-12345678
```

出力:

```
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
```

例 2: この例では、「Name」タグを持つスナップショットを記述します。

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" }
```

例 3: この例では、「Name」タグと値「TestValue」を持つスナップショットを記述します。

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" -and
                        $_.Tags.Value -eq "TestValue" }
```

例 4: この例では、すべてのスナップショットを記述します。

```
Get-EC2Snapshot -Owner self
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeSnapshots](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

スナップショットの状態を表示します。

```
async fn show_state(client: &Client, id: &str) -> Result<(), Error> {
    let resp = client
        .describe_snapshots()
        .filters(Filter::builder().name("snapshot-id").values(id).build())
        .send()
        .await?;

    println!(
        "State: {}",
        resp.snapshots().first().unwrap().state().unwrap().as_ref()
    );

    Ok(())
}
```

```
async fn show_snapshots(client: &Client) -> Result<(), Error> {
    // "self" represents your account ID.
    // You can list the snapshots for any account by replacing
    // "self" with that account ID.
    let resp = client.describe_snapshots().owner_ids("self").send().await?;
    let snapshots = resp.snapshots();
    let length = snapshots.len();
```

```
for snapshot in snapshots {
    println!(
        "ID:          {}",
        snapshot.snapshot_id().unwrap_or_default()
    );
    println!(
        "Description: {}",
        snapshot.description().unwrap_or_default()
    );
    println!("State:          {}", snapshot.state().unwrap().as_ref());
    println!();
}

println!();
println!("Found {} snapshot(s)", length);
println!();

Ok(())
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[DescribeSnapshots](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeSpotDatafeedSubscription** を使用する

次のサンプルコードは、DescribeSpotDatafeedSubscription を使用方法を説明しています。

CLI

AWS CLI

アカウントのスポットインスタンスデータフィードサブスクリプションを記述するには
このコマンド例では、アカウントのデータフィードを記述します。

コマンド:

```
aws ec2 describe-spot-datafeed-subscription
```

出力:

```
{
  "SpotDatafeedSubscription": {
    "OwnerId": "123456789012",
    "Prefix": "spotdata",
    "Bucket": "amzn-s3-demo-bucket",
    "State": "Active"
  }
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeSpotDatafeedSubscription](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、スポットインスタンスのデータフィードについて記述します。

```
Get-EC2SpotDatafeedSubscription
```

出力:

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeSpotDatafeedSubscription](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、スポットインスタンスのデータフィードについて記述します。

```
Get-EC2SpotDatafeedSubscription
```

出力:

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeSpotDatafeedSubscription](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください [AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeSpotFleetInstances** を使用する

次のサンプルコードは、DescribeSpotFleetInstances を使用する方法を説明しています。

CLI

AWS CLI

スポットフリートに関連付けられたスポットインスタンスを記述するには

このコマンド例では、指定されたスポットフリートに関連付けられたスポットインスタンスを一覧表示します。

コマンド:

```
aws ec2 describe-spot-fleet-instances --spot-fleet-request-id sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
```

出力:

```
{
  "ActiveInstances": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "InstanceType": "m3.medium",
      "SpotInstanceRequestId": "sir-08b93456"
```

```
    },
    ...
  ],
  "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE"
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeSpotFleetInstances](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたスポットフリートリクエストに関連付けられたインスタンスについて記述します。

```
Get-EC2SpotFleetInstance -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
```

出力:

InstanceId	InstanceType	SpotInstanceRequestId
-----	-----	-----
i-f089262a	c3.large	sir-12345678
i-7e8b24a4	c3.large	sir-87654321

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeSpotFleetInstances](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたスポットフリートリクエストに関連付けられたインスタンスについて記述します。

```
Get-EC2SpotFleetInstance -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
```

出力:

InstanceId	InstanceType	SpotInstanceRequestId

i-f089262a	c3.large	sir-12345678
i-7e8b24a4	c3.large	sir-87654321

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeSpotFleetInstances](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で `DescribeSpotFleetRequestHistory` を使用する

次のサンプルコードは、`DescribeSpotFleetRequestHistory` を使用方法を説明しています。

CLI

AWS CLI

スポットフリートの履歴を記述するには

このコマンド例では、指定された時刻から始まる、指定されたスポットフリートの履歴を返します。

コマンド:

```
aws ec2 describe-spot-fleet-request-history --spot-fleet-request-id sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE --start-time 2015-05-26T00:00:00Z
```

次の出力例は、スポットフリートの 2 つのスポットインスタンスの正常な起動を示しています。

出力:

```
{
  "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
+53MNlR0YcXAkp0xF1fKf91yVxSExmbtma3awYxMFzNA663ZskT0AhtJ6TCb2Z8bQC2EnZgyELbymtWPfpZ1ZbauV
+P+TfG1WxWWB/Vr5dk5d4LfdgA/DRAHUrYgxzrEXAMPLE=",
  "StartTime": "2015-05-26T00:00:00Z"
}

```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[DescribeSpotFleetRequestHistory](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたスポットフリートリクエストの履歴を記述します。

```
Get-EC2SpotFleetRequestHistory -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -StartTime 2015-12-26T00:00:00Z
```

出力:

```
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
```

出力:

EventInformation	EventType	Timestamp
-----	-----	-----
Amazon.EC2.Model.EventInformation	fleetRequestChange	12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation	fleetRequestChange	12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation	fleetRequestChange	12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation	launched	12/26/2015 8:25:34 AM
Amazon.EC2.Model.EventInformation	launched	12/26/2015 8:25:05 AM

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeSpotFleetRequestHistory](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたスポットフリートリクエストの履歴を記述します。

```
Get-EC2SpotFleetRequestHistory -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -StartTime 2015-12-26T00:00:00Z
```

出力:

```
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
```

出力:

EventInformation	EventType	Timestamp
-----	-----	-----
Amazon.EC2.Model.EventInformation	fleetRequestChange	12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation	fleetRequestChange	12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation	fleetRequestChange	12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation	launched	12/26/2015 8:25:34 AM
Amazon.EC2.Model.EventInformation	launched	12/26/2015 8:25:05 AM

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeSpotFleetRequestHistory](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeSpotFleetRequests** を使用する

次のサンプルコードは、DescribeSpotFleetRequests を使用する方法を説明しています。

CLI

AWS CLI

スポットフリートリクエストを記述するには

この例では、すべてのスポットフリートリクエストを記述します。

コマンド:

```
aws ec2 describe-spot-fleet-requests
```

出力:

```
{
  "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"
  }
]
}

```

スポットフリートリクエストを記述するには

この例では、指定されたスポットフリートリクエストを記述します。

コマンド:

```
aws ec2 describe-spot-fleet-requests --spot-fleet-request-ids sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
```

出力:

```

{
  "SpotFleetRequestConfigs": [
    {
      "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE",

```

```

    "SpotFleetRequestConfig": {
      "TargetCapacity": 20,
      "LaunchSpecifications": [
        {
          "EbsOptimized": false,
          "NetworkInterfaces": [
            {
              "SubnetId": "subnet-a61dafcf",
              "DeviceIndex": 0,
              "DeleteOnTermination": false,
              "AssociatePublicIpAddress": true,
              "SecondaryPrivateIpAddressCount": 0
            }
          ],
          "InstanceType": "cc2.8xlarge",
          "ImageId": "ami-1a2b3c4d"
        },
        {
          "EbsOptimized": false,
          "NetworkInterfaces": [
            {
              "SubnetId": "subnet-a61dafcf",
              "DeviceIndex": 0,
              "DeleteOnTermination": false,
              "AssociatePublicIpAddress": true,
              "SecondaryPrivateIpAddressCount": 0
            }
          ],
          "InstanceType": "r3.8xlarge",
          "ImageId": "ami-1a2b3c4d"
        }
      ],
      "SpotPrice": "0.05",
      "IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role"
    },
    "SpotFleetRequestState": "active"
  }
}

```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[DescribeSpotFleetRequests](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたスポットフリートリクエストを記述します。

```
Get-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE | format-list
```

出力:

```
ConfigData           : Amazon.EC2.Model.SpotFleetRequestConfigData
CreateTime           : 12/26/2015 8:23:33 AM
SpotFleetRequestId   : sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
SpotFleetRequestState : active
```

例 2: この例では、すべてのスポットフリートリクエストを記述します。

```
Get-EC2SpotFleetRequest
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeSpotFleetRequests](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたスポットフリートリクエストを記述します。

```
Get-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE | format-list
```

出力:

```
ConfigData           : Amazon.EC2.Model.SpotFleetRequestConfigData
CreateTime           : 12/26/2015 8:23:33 AM
SpotFleetRequestId   : sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
SpotFleetRequestState : active
```

例 2: この例では、すべてのスポットフリートリクエストを記述します。

```
Get-EC2SpotFleetRequest
```


- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeSpotFleetRequests](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeSpotInstanceRequests** を使用する

次のサンプルコードは、DescribeSpotInstanceRequests を使用する方法を説明しています。

CLI

AWS CLI

例 1: スポットインスタンスリクエストを記述するには

次の describe-spot-instance-requests の例では、指定したスポットインスタンスについて説明します。

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

出力:

```
{  
  "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-0e54a519c999adbdbd",
            "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"
}
]
}

```

例 2: フィルターに基づいてスポットインスタンスリクエストを記述するには

次の `describe-spot-instance-requests` の例では、フィルターを使用して、指定したアベイラビリティゾーンにある指定したインスタントタイプを持つスポットインスタンスリクエストに結果の範囲を絞り込みます。この例では、`--query` パラメータを使用してインスタンス ID のみを表示します。

```
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
```

出力:

```
i-057750d42936e468a
i-001efd250faaa6ffa
i-027552a73f021f3bd
...
```

フィルターの使用の詳細については、「Amazon Elastic Compute Cloud ユーザーガイド」の「[Listing and filtering your resources](#)」を参照してください。

例 3: タグに基づいてスポットインスタンスリクエストを記述するには

次の `describe-spot-instance-requests` の例では、タグフィルターを使用して、タグ `cost-center=cc123` を含むスポットインスタンスリクエストに結果の範囲を絞り込みます。

```
aws ec2 describe-spot-instance-requests \
  --filters Name=tag:cost-center,Values=cc123
```

`describe-spot-instance-requests` の出力例については、例 1 を参照してください。

タグフィルターを使用するその他の例については、「Amazon EC2 ユーザーガイド」の「[タグの使用](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeSpotInstanceRequests](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定したスポットインスタンスを記述します。

```
Get-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

出力:

```
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
```

例 2: この例では、すべてのスポットインスタンスリクエストを記述します。

```
Get-EC2SpotInstanceRequest
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeSpotInstanceRequests](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定したスポットインスタンスを記述します。

```
Get-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

出力:

```
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
```

例 2: この例では、すべてのスポットインスタンスリクエストを記述します。

```
Get-EC2SpotInstanceRequest
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeSpotInstanceRequests](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください [AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeSpotPriceHistory** を使用する

次のサンプルコードは、DescribeSpotPriceHistory を使用方法を説明しています。

CLI

AWS CLI

スポット料金履歴を記述するには

このコマンド例では、1 月の特定の日について m1.xlarge インスタンスのスポット料金履歴を返します。

コマンド:

```
aws ec2 describe-spot-price-history --instance-types m1.xlarge --start-time 2014-01-06T07:08:09 --end-time 2014-01-06T08:09:10
```

出力:

```
{
  "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"
    },
    ...
  ]
}
```

Linux/UNIX Amazon VPC のスポット料金履歴を記述するには

このコマンド例では、1 月の特定の日について m1.xlarge Linux/UNIX Amazon VPC インスタンスのスポット料金履歴を返します。

コマンド:

```
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
```

出力:

```
{
  "SpotPriceHistory": [
    {
      "Timestamp": "2014-01-06T04:32:53.000Z",
      "ProductDescription": "Linux/UNIX (Amazon VPC)",
      "InstanceType": "m1.xlarge",
      "SpotPrice": "0.080000",
      "AvailabilityZone": "us-west-1a"
    },
    {
      "Timestamp": "2014-01-05T11:28:26.000Z",
      "ProductDescription": "Linux/UNIX (Amazon VPC)",
      "InstanceType": "m1.xlarge",
      "SpotPrice": "0.080000",
      "AvailabilityZone": "us-west-1c"
    }
  ]
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeSpotPriceHistory](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたインスタンスタイプとアベイラビリティゾーンのスبット料金履歴から直近 10 件のエントリを取得します。-AvailabilityZone パラメータに指定された値は、コマンドレットの -Region パラメータ (例に示されていない) に指定されたリージョン値に対して有効であるか、シェルでデフォルトとして設定されている必要があることに注意してください。このコマンド例では、「us-west-2」のデフォルトリージョンが環境で設定されていることを前提としています。

```
Get-EC2SpotPriceHistory -InstanceType c3.large -AvailabilityZone us-west-2a -
MaxResult 10
```

出力:

```
AvailabilityZone    : us-west-2a
```

```

InstanceType      : c3.large
Price             : 0.017300
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp         : 12/25/2015 7:39:49 AM

AvailabilityZone   : us-west-2a
InstanceType      : c3.large
Price             : 0.017200
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp         : 12/25/2015 7:38:29 AM

AvailabilityZone   : us-west-2a
InstanceType      : c3.large
Price             : 0.017300
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp         : 12/25/2015 6:57:13 AM
...

```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeSpotPriceHistory](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたインスタンスタイプとアベイラビリティゾーンのスبット料金履歴から直近 10 件のエントリを取得します。-AvailabilityZone パラメータに指定された値は、コマンドレットの -Region パラメータ (例に示されていない) に指定されたリージョン値に対して有効であるか、シェルでデフォルトとして設定されている必要があることに注意してください。このコマンド例では、「us-west-2」のデフォルトリージョンが環境で設定されていることを前提としています。

```
Get-EC2SpotPriceHistory -InstanceType c3.large -AvailabilityZone us-west-2a -
MaxResult 10
```

出力:

```

AvailabilityZone   : us-west-2a
InstanceType      : c3.large
Price             : 0.017300
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp         : 12/25/2015 7:39:49 AM

AvailabilityZone   : us-west-2a

```



```
InstanceType      : c3.large
Price             : 0.017200
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp         : 12/25/2015 7:38:29 AM

AvailabilityZone   : us-west-2a
InstanceType      : c3.large
Price             : 0.017300
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp         : 12/25/2015 6:57:13 AM
...
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeSpotPriceHistory](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DescribeSubnets**で を使用する

次のサンプルコードは、DescribeSubnets を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [レジリエントなサービスの構築と管理](#)
- [Amazon VPC の使用を開始する](#)
- [Transit Gateway を開始する](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Get all the subnets for a Vpc in a set of availability zones.
/// </summary>
/// <param name="vpcId">The Id of the Vpc.</param>
/// <param name="availabilityZones">The list of availability zones.</param>
/// <returns>The collection of subnet objects.</returns>
public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
List<string> availabilityZones)
{
    try
    {
        var subnets = new List<Subnet>();
        var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
            new DescribeSubnetsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new("vpc-id", new List<string>() { vpcId }),
                    new("availability-zone", availabilityZones),
                    new("default-for-az", new List<string>() { "true" })
                }
            });

        // Get the entire list using the paginator.
        await foreach (var subnet in subnetPaginator.Subnets)
        {
            subnets.Add(subnet);
        }

        return subnets;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
        {
            _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId}
does not exist.");
        }

        throw;
    }
    catch (Exception ex)
    {

```

```
        _logger.LogError(ex, $"An error occurred while describing the  
subnets.: {ex.Message}");  
        throw;  
    }  
}
```

- APIの詳細については、「AWS SDK for .NET API リファレンス」の「[DescribeSubnets](#)」を参照してください。

CLI

AWS CLI

例 1: すべてのサブネットを説明するには

次の describe-subnets の例では、サブネットの詳細を示します。

```
aws ec2 describe-subnets
```

出力:

```
{  
  "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": 4096,
        "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
        }
    }
]
}

```

詳細については、「AWS VPC ユーザーガイド」で [VPC とサブネットの使用方法](#) を参照してください。

例 2: 特定の VPC のサブネットを説明するには

次の `describe-subnets` 例では、フィルターを使用して、指定した VPC のサブネットに関する詳細を取得します。

```
aws ec2 describe-subnets \
  --filters "Name=vpc-id,Values=vpc-3EXAMPLE"
```

出力:

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

```
}
```

詳細については、「AWS VPC ユーザーガイド」で [VPC とサブネットの使用方法](#) を参照してください。

例 3: 特定のタグを持つサブネットを説明するには

次の describe-subnets の例では、フィルターを使用してタグ CostCenter=123 付きのサブネットの詳細を取得し、--query パラメータを使用してこのタグが付いたサブネットのサブネット ID を表示します。

```
aws ec2 describe-subnets \
  --filters "Name=tag:CostCenter,Values=123" \
  --query "Subnets[*].SubnetId" \
  --output text
```

出力:

```
subnet-0987a87c8b37348ef
subnet-02a95061c45f372ee
subnet-03f720e7de2788d73
```

詳細については、「Amazon VPC ユーザーガイド」で [VPC とサブネットの使用方法](#) を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeSubnets](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
const client = new EC2Client({});
```

```
const { Subnets } = await client.send(  
  new DescribeSubnetsCommand({  
    Filters: [  
      { Name: "vpc-id", Values: [state.defaultVpc] },  
      { Name: "availability-zone", Values: state.availabilityZoneNames },  
      { Name: "default-for-az", Values: ["true"] },  
    ],  
  })),  
);
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[DescribeSubnets](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定したサブネットを記述します。

```
Get-EC2Subnet -SubnetId subnet-1a2b3c4d
```

出力:

```
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
```

例 2: この例では、すべてのサブネットを記述します。

```
Get-EC2Subnet
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeSubnets](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定したサブネットを記述します。

```
Get-EC2Subnet -SubnetId subnet-1a2b3c4d
```

出力:

```
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
```

例 2: この例では、すべてのサブネットを記述します。

```
Get-EC2Subnet
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeSubnets](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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



```

def __init__(
    self,
    resource_prefix: str,
    inst_type: str,
    ami_param: str,
    autoscaling_client: boto3.client,
    ec2_client: boto3.client,
    ssm_client: boto3.client,
    iam_client: boto3.client,
):
    """
    Initializes the AutoScaler class with the necessary parameters.

    :param resource_prefix: The prefix for naming AWS resources that are
    created by this class.
    :param inst_type: The type of EC2 instance to create, such as t3.micro.
    :param ami_param: The Systems Manager parameter used to look up the AMI
    that is created.
    :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
    :param ec2_client: A Boto3 EC2 client.
    :param ssm_client: A Boto3 Systems Manager client.
    :param iam_client: A Boto3 IAM client.
    """
    self.inst_type = inst_type
    self.ami_param = ami_param
    self.autoscaling_client = autoscaling_client
    self.ec2_client = ec2_client
    self.ssm_client = ssm_client
    self.iam_client = iam_client
    sts_client = boto3.client("sts")
    self.account_id = sts_client.get_caller_identity()["Account"]

    self.key_pair_name = f"{resource_prefix}-key-pair"
    self.launch_template_name = f"{resource_prefix}-template-"
    self.group_name = f"{resource_prefix}-group"

    # Happy path
    self.instance_policy_name = f"{resource_prefix}-pol"
    self.instance_role_name = f"{resource_prefix}-role"
    self.instance_profile_name = f"{resource_prefix}-prof"

    # Failure mode
    self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
    self.bad_creds_role_name = f"{resource_prefix}-bc-role"

```

```

self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
    """
    Gets the default subnets in a VPC for a specified list of Availability
    Zones.

    :param vpc_id: The ID of the VPC to look up.
    :param zones: The list of Availability Zones to look up.
    :return: The list of subnets found.
    """
    # Ensure that 'zones' is a list, even if None is passed
    if zones is None:
        zones = []
    try:
        paginator = self.ec2_client.get_paginator("describe_subnets")
        page_iterator = paginator.paginate(
            Filters=[
                {"Name": "vpc-id", "Values": [vpc_id]},
                {"Name": "availability-zone", "Values": zones},
                {"Name": "default-for-az", "Values": ["true"]},
            ]
        )

        subnets = []
        for page in page_iterator:
            subnets.extend(page["Subnets"])

        log.info("Found %s subnets for the specified zones.", len(subnets))
        return subnets
    except ClientError as err:
        log.error(
            f"Failed to retrieve subnets for VPC '{vpc_id}' in zones
{zones}."
        )
        error_code = err.response["Error"]["Code"]
        if error_code == "InvalidVpcID.NotFound":
            log.error(
                "The specified VPC ID does not exist. "
                "Please check the VPC ID and try again."
            )
        # Add more error-specific handling as needed

```

```
log.error(f"Full error:\n\t{err}")
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[DescribeSubnets](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeTags** を使用する

次のサンプルコードは、DescribeTags を使用する方法を説明しています。

CLI

AWS CLI

例 1: 1 つのリソースのタグをすべて記述するには

次の describe-tags の例では、指定したインスタンスのタグについて説明します。

```
aws ec2 describe-tags \
  --filters "Name=resource-id,Values=i-1234567890abcdef8"
```

出力:

```
{
  "Tags": [
    {
      "ResourceType": "instance",
      "ResourceId": "i-1234567890abcdef8",
      "Value": "Test",
      "Key": "Stack"
    },
    {
      "ResourceType": "instance",
      "ResourceId": "i-1234567890abcdef8",
      "Value": "Beta Server",
      "Key": "Name"
    }
  ]
}
```

```
]
}
```

例 2: リソースタイプのタグをすべて記述するには

次の describe-tags の例では、ボリュームのタグについて説明します。

```
aws ec2 describe-tags \
  --filters "Name=resource-type,Values=volume"
```

出力:

```
{
  "Tags": [
    {
      "ResourceType": "volume",
      "ResourceId": "vol-1234567890abcdef0",
      "Value": "Project1",
      "Key": "Purpose"
    },
    {
      "ResourceType": "volume",
      "ResourceId": "vol-049df61146c4d7901",
      "Value": "Logs",
      "Key": "Purpose"
    }
  ]
}
```

例 3: タグをすべて記述するには

次の describe-tags の例では、リソースのタグについて説明します。

```
aws ec2 describe-tags
```

例 4: タグキーに基づいてリソースのタグを記述するには

次の describe-tags の例では、キー Stack を持つタグのあるリソースのタグについて説明します。

```
aws ec2 describe-tags \
```

```
--filters Name=key,Values=Stack
```

出力:

```
{
  "Tags": [
    {
      "ResourceType": "volume",
      "ResourceId": "vol-027552a73f021f3b",
      "Value": "Production",
      "Key": "Stack"
    },
    {
      "ResourceType": "instance",
      "ResourceId": "i-1234567890abcdef8",
      "Value": "Test",
      "Key": "Stack"
    }
  ]
}
```

例 5: タグキーとタグ値に基づいてリソースのタグを記述するには

次の describe-tags の例では、タグ Stack=Test のあるリソースのタグについて説明します。

```
aws ec2 describe-tags \
  --filters Name=key,Values=Stack Name=value,Values=Test
```

出力:

```
{
  "Tags": [
    {
      "ResourceType": "image",
      "ResourceId": "ami-3ac336533f021f3bd",
      "Value": "Test",
      "Key": "Stack"
    },
    {
      "ResourceType": "instance",
      "ResourceId": "i-1234567890abcdef8",

```

```
        "Value": "Test",
        "Key": "Stack"
    }
]
}
```

次の describe-tags の例では、代替構文を使用して、タグ Stack=Test を持つリソースについて説明します。

```
aws ec2 describe-tags \
  --filters "Name=tag:Stack,Values=Test"
```

次の describe-tags の例では、キー Purpose を持つものの値を持っていないタグのある、すべてのインスタンスのタグについて説明します。

```
aws ec2 describe-tags \
  --filters "Name=resource-type,Values=instance" "Name=key,Values=Purpose" "Name=value,Values="
```

出力:

```
{
  "Tags": [
    {
      "ResourceType": "instance",
      "ResourceId": "i-1234567890abcdef5",
      "Value": null,
      "Key": "Purpose"
    }
  ]
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeTags](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、リソースタイプ「image」のタグを取得します。

```
Get-EC2Tag -Filter @{"Name"="resource-type";Values="image"}
```

出力:

Key	ResourceId	ResourceType	Value
---	-----	-----	-----
Name	ami-0a123b4ccb567a8ea	image	Win7-Imported
auto-delete	ami-0a123b4ccb567a8ea	image	never

例 2: この例では、すべてのリソースのすべてのタグを取得し、リソースタイプ別にグループ化します。

```
Get-EC2Tag | Group-Object resourcetype
```

出力:

Count	Name	Group
-----	----	-----
9	subnet	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription...}
53	instance	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription...}
3	route-table	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
5	security-group	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription...}
30	volume	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription...}
1	internet-gateway	{Amazon.EC2.Model.TagDescription}
3	network-interface	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
4	elastic-ip	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
1	dhcp-options	{Amazon.EC2.Model.TagDescription}
2	image	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}

```
3 vpc                                     {Amazon.EC2.Model.TagDescription,
Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
```

例 3: この例では、指定されたリージョンでタグ「auto-delete」と値「no」を持つすべてのリソースを表示します。

```
Get-EC2Tag -Region eu-west-1 -Filter @{{Name="tag:auto-delete";Values="no"}}
```

出力:

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

例 4: この例では、タグ「auto-delete」と値「no」を持つすべてのリソースを取得し、次のパイプでさらにフィルターして「instance」リソースタイプのみを解析し、最終的に各インスタンスリソースに対して、そのインスタンス ID 自体を値とする「ThisInstance」タグを作成します。

```
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}}
```

例 5: この例では、すべてのインスタンスリソースのタグと「Name」キーを取得し、テーブル形式で表示します。

```
Get-EC2Tag -Filter @{{Name="resource-
type";Values="instance"},@{{Name="key";Values="Name"}} | Select-Object ResourceId,
@{{Name="Name-Tag";Expression={$PSItem.Value}}} | Format-Table -AutoSize
```

出力:

ResourceId	Name-Tag
-----	-----
i-012e3cb4df567e1aa	jump1
i-01c23a45d6fc7a89f	repro-3

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeTags](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、リソースタイプ「image」のタグを取得します。

```
Get-EC2Tag -Filter @{"Name"="resource-type";Values="image"}
```

出力:

Key	ResourceId	ResourceType	Value
---	-----	-----	-----
Name	ami-0a123b4ccb567a8ea	image	Win7-Imported
auto-delete	ami-0a123b4ccb567a8ea	image	never

例 2: この例では、すべてのリソースのすべてのタグを取得し、リソースタイプ別にグループ化します。

```
Get-EC2Tag | Group-Object resourcetype
```

出力:

Count	Name	Group
-----	-----	-----
9	subnet	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription...}
53	instance	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription...}
3	route-table	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
5	security-group	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription...}
30	volume	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription...}
1	internet-gateway	{Amazon.EC2.Model.TagDescription}
3	network-interface	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}

```

4 elastic-ip          {Amazon.EC2.Model.TagDescription,
Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
Amazon.EC2.Model.TagDescription}
1 dhcp-options        {Amazon.EC2.Model.TagDescription}
2 image               {Amazon.EC2.Model.TagDescription,
Amazon.EC2.Model.TagDescription}
3 vpc                 {Amazon.EC2.Model.TagDescription,
Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}

```

例 3: この例では、指定されたリージョンでタグ「auto-delete」と値「no」を持つすべてのリソースを表示します。

```
Get-EC2Tag -Region eu-west-1 -Filter @{"Name"="tag:auto-delete";Values="no"}
```

出力:

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

例 4: この例では、タグ「auto-delete」と値「no」を持つすべてのリソースを取得し、次のパイプでさらにフィルターして「instance」リソースタイプのみを解析し、最終的に各インスタンスリソースに対して、そのインスタンス ID 自体を値とする「ThisInstance」タグを作成します。

```

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}}

```

例 5: この例では、すべてのインスタンスリソースのタグと「Name」キーを取得し、テーブル形式で表示します。

```

Get-EC2Tag -Filter @{"Name"="resource-
type";Values="instance"},@{"Name"="key";Values="Name"} | Select-Object ResourceId,
@{"Name"="Name-Tag";Expression={$PSItem.Value}} | Format-Table -AutoSize

```

出力:

ResourceId	Name-Tag
-----	-----
i-012e3cb4df567e1aa	jump1
i-01c23a45d6fc7a89f	repro-3

例 6: この例では、実際に EC2 タグを取得することなく、DryRun パラメータを使用して EC2 タグ取得のアクセス許可を検証します。注: 成功すると例外がスローされますが、これは正常な動作です。

```
Get-EC2Tag -DryRun $true
```

出力:

```
Get-EC2Tag: Request would have succeeded, but DryRun flag is set.
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[DescribeTags](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeVolumeAttribute** を使用する

次のサンプルコードは、DescribeVolumeAttribute を使用する方法を説明しています。

CLI

AWS CLI

ボリューム属性を記述するには

このコマンド例では、ID vol-049df61146c4d7901 を持つボリュームの autoEnableIo 属性を記述します。

コマンド:

```
aws ec2 describe-volume-attribute --volume-id vol-049df61146c4d7901 --  
attribute autoEnableIO
```

出力:

```
{
  "AutoEnableIO": {
    "Value": false
  },
  "VolumeId": "vol-049df61146c4d7901"
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeVolumeAttribute](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたボリュームの指定された属性を記述します。

```
Get-EC2VolumeAttribute -VolumeId vol-12345678 -Attribute AutoEnableIO
```

出力:

AutoEnableIO	ProductCodes	VolumeId
-----	-----	-----
False	{}	vol-12345678

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeVolumeAttribute](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたボリュームの指定された属性を記述します。

```
Get-EC2VolumeAttribute -VolumeId vol-12345678 -Attribute AutoEnableIO
```

出力:

AutoEnableIO	ProductCodes	VolumeId
-----	-----	-----
False	{}	vol-12345678

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeVolumeAttribute](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeVolumeStatus** を使用する

次のサンプルコードは、DescribeVolumeStatus を使用する方法を説明しています。

CLI

AWS CLI

単一ボリュームのステータスを記述するには

このコマンド例では、ボリューム vol-1234567890abcdef0 のステータスを記述します。

コマンド:

```
aws ec2 describe-volume-status --volume-ids vol-1234567890abcdef0
```

出力:

```
{
  "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": []
    }
]
```

障害のあるボリュームのステータスを記述するには

このコマンド例では、障害のあるすべてのボリュームのステータスを記述します。この例では、障害のあるボリュームはありません。

コマンド:

```
aws ec2 describe-volume-status --filters Name=volume-  
status.status,Values=impaired
```

出力:

```
{
  "VolumeStatuses": []
}
```

ステータスチェックに失敗したボリュームがある場合 (ステータスに障害がある場合)、
「Amazon EC2 ユーザーガイド」の「Working with an Impaired Volume」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeVolumeStatus](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたボリュームのステータスを記述します。

```
Get-EC2VolumeStatus -VolumeId vol-12345678
```

出力:

```

Actions          : {}
AvailabilityZone  : us-west-2a
Events           : {}
VolumeId         : vol-12345678
VolumeStatus     : Amazon.EC2.Model.VolumeStatusInfo

```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus
```

出力:

Details	Status
-----	-----
{io-enabled, io-performance}	ok

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus.Details
```

出力:

Name	Status
----	-----
io-enabled	passed
io-performance	not-applicable

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeVolumeStatus](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたボリュームのステータスを記述します。

```
Get-EC2VolumeStatus -VolumeId vol-12345678
```

出力:

```

Actions          : {}
AvailabilityZone  : us-west-2a
Events           : {}
VolumeId         : vol-12345678
VolumeStatus     : Amazon.EC2.Model.VolumeStatusInfo

```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus
```

出力:

Details	Status
-----	-----
{io-enabled, io-performance}	ok

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus.Details
```

出力:

Name	Status
----	-----
io-enabled	passed
io-performance	not-applicable

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeVolumeStatus](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeVolumes** を使用する

次のサンプルコードは、DescribeVolumes を使用する方法を説明しています。

CLI

AWS CLI

例 1: ボリュームを記述するには

次の describe-volumes の例では、現在のリージョンで指定したボリュームについて説明します。

```
aws ec2 describe-volumes \
```



```
--volume-ids vol-049df61146c4d7901 vol-1234567890abcdef0
```

出力:

```
{
  "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
    }
  ]
}
```

例 2: 特定のインスタンスにアタッチされているボリュームを記述するには

次の `describe-volumes` の例では、指定したインスタンスにアタッチされているものと、インスタンスの終了時に削除するように設定されているものの両方に該当するすべてのボリュームについて説明します。

```
aws ec2 describe-volumes \
  --region us-east-1 \
  --filters Name=attachment.instance-  
id,Values=i-1234567890abcdef0 Name=attachment.delete-on-termination,Values=true
```

`describe-volumes` の出力例については、例 1 を参照してください。

例 3: 特定のアベイラビリティゾーンで使用可能なボリュームを記述するには

次の `describe-volumes` の例では、ステータスが `available` であり、指定したアベイラビリティゾーンにあるすべてのボリュームについて説明します。

```
aws ec2 describe-volumes \
  --filters Name=status,Values=available Name=availability-zone,Values=us-  
east-1a
```

`describe-volumes` の出力例については、例 1 を参照してください。

例 4: タグに基づいてボリュームを記述するには

次の `describe-volumes` の例では、タグキー `Name` を持ち、`Test` で始まる値を持つすべてのボリュームについて説明します。その後、出力は、ボリュームのタグと ID のみを表示するクエリでフィルタリングされます。

```
aws ec2 describe-volumes \
  --filters Name=tag:Name,Values=Test* \  
--query "Volumes[*].{ID:VolumeId,Tag:Tags}"
```

出力:

```
[
  {
    "Tag": [
      {
        "Value": "Test2",
```

```
        "Key": "Name"
      }
    ],
    "ID": "vol-1234567890abcdef0"
  },
  {
    "Tag": [
      {
        "Value": "Test1",
        "Key": "Name"
      }
    ],
    "ID": "vol-049df61146c4d7901"
  }
]
```

タグフィルターを使用するその他の例については、「Amazon EC2 ユーザーガイド」の「[タグの使用法](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeVolumes](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定した EBS ボリュームを記述します。

```
Get-EC2Volume -VolumeId vol-12345678
```

出力:

```
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
```

例 2: この例では、ステータスが「使用可能」の EBS ボリュームを記述します。

```
Get-EC2Volume -Filter @{ Name="status"; Values="available" }
```

出力:

```
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
...
```

例 3: この例では、すべての EBS ボリュームを記述します。

```
Get-EC2Volume
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeVolumes](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定した EBS ボリュームを記述します。

```
Get-EC2Volume -VolumeId vol-12345678
```

出力:

```
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
```

例 2: この例では、ステータスが「使用可能」の EBS ボリュームを記述します。

```
Get-EC2Volume -Filter @{ Name="status"; Values="available" }
```

出力:

```
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
...
```

例 3: この例では、すべての EBS ボリュームを記述します。

```
Get-EC2Volume
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeVolumes](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeVpcAttribute** を使用する

次のサンプルコードは、`DescribeVpcAttribute` を使用する方法を説明しています。

CLI

AWS CLI

`enableDnsSupport` 属性を記述するには

この例では、`enableDnsSupport` 属性を記述します。この属性は、VPC に対して DNS 解決が有効になっているかどうかを示します。この属性が `true` の場合、Amazon DNS サーバーはインスタンスの DNS ホスト名を対応する IP アドレスに解決します。それ以外の場合は解決しません。

コマンド:

```
aws ec2 describe-vpc-attribute --vpc-id vpc-a01106c2 --attribute enableDnsSupport
```

出力:

```
{
  "VpcId": "vpc-a01106c2",
  "EnableDnsSupport": {
    "Value": true
  }
}
```

`enableDnsHostnames` 属性を記述するには

この例では、`enableDnsHostnames` 属性を記述します。この属性は、VPC 内に起動されるインスタンスが DNS ホスト名を取得するかどうかを示します。この属性が `true` の場合、VPC 内のインスタンスは DNS ホスト名を取得します。それ以外の場合は取得しません。

コマンド:

```
aws ec2 describe-vpc-attribute --vpc-id vpc-a01106c2 --
attribute enableDnsHostnames
```

出力:

```
{
  "VpcId": "vpc-a01106c2",
  "EnableDnsHostnames": {
    "Value": true
  }
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeVpcAttribute](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、「enableDnsSupport」属性を記述します。

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsSupport
```

出力:

```
EnableDnsSupport
-----
True
```

例 2: この例では、「enableDnsHostnames」属性を記述します。

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsHostnames
```

出力:

```
EnableDnsHostnames
-----
True
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeVpcAttribute](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、「enableDnsSupport」属性を記述します。

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsSupport
```

出力:

```
EnableDnsSupport
-----
True
```

例 2: この例では、「enableDnsHostnames」属性を記述します。

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsHostnames
```

出力:

```
EnableDnsHostnames
-----
True
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeVpcAttribute](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeVpcClassicLink** を使用する

次のサンプルコードは、DescribeVpcClassicLink を使用する方法を説明しています。

CLI

AWS CLI

VPC の ClassicLink ステータスを記述するには

この例では、vpc-88888888 の ClassicLink ステータスを一覧表示します。

コマンド:


```
aws ec2 describe-vpc-classic-link --vpc-id vpc-88888888
```

出力:

```
{
  "Vpcs": [
    {
      "ClassicLinkEnabled": true,
      "VpcId": "vpc-88888888",
      "Tags": [
        {
          "Value": "classiclinkvpc",
          "Key": "Name"
        }
      ]
    }
  ]
}
```

この例では、Classiclink で有効になっている VPCs のみを一覧表示します (is-classic-link-enabled のフィルター値は true に設定されています)。

コマンド:

```
aws ec2 describe-vpc-classic-link --filter "Name=is-classic-link-enabled,Values=true"
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeVpcClassicLink](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: 上記の例では、リージョンの ClassicLinkEnabled 状態を持つすべての VPC を返します。

```
Get-EC2VpcClassicLink -Region eu-west-1
```

出力:

ClassicLinkEnabled	Tags	VpcId
-----	----	-----
False	{Name}	vpc-0fc1ff23f45b678eb
False	{}	vpc-01e23c4a5d6db78e9
False	{Name}	vpc-0123456b078b9d01f
False	{}	vpc-12cf3b4f
False	{Name}	vpc-0b12d3456a7e8901d

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeVpcClassicLink](#)」を参照してください。

Tools for PowerShell V5

例 1: 上記の例では、リージョンの ClassicLinkEnabled 状態を持つすべての VPC を返します。

```
Get-EC2VpcClassicLink -Region eu-west-1
```

出力:

ClassicLinkEnabled	Tags	VpcId
-----	----	-----
False	{Name}	vpc-0fc1ff23f45b678eb
False	{}	vpc-01e23c4a5d6db78e9
False	{Name}	vpc-0123456b078b9d01f
False	{}	vpc-12cf3b4f
False	{Name}	vpc-0b12d3456a7e8901d

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeVpcClassicLink](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeVpcClassicLinkDnsSupport** を使用する

次のサンプルコードは、DescribeVpcClassicLinkDnsSupport を使用する方法を説明しています。

CLI

AWS CLI

VPC の ClassicLink DNS サポートを記述するには

この例では、すべての VPC の ClassicLink DNS サポートステータスを記述します。

コマンド:

```
aws ec2 describe-vpc-classic-link-dns-support
```

出力:

```
{
  "Vpcs": [
    {
      "VpcId": "vpc-88888888",
      "ClassicLinkDnsSupported": true
    },
    {
      "VpcId": "vpc-1a2b3c4d",
      "ClassicLinkDnsSupported": false
    }
  ]
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeVpcClassicLinkDnsSupport](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、リージョン eu-west-1 の VPC の ClassicLink DNS サポートステータスを記述します。

```
Get-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

出力:

```
ClassicLinkDnsSupported VpcId
-----
False                  vpc-0b12d3456a7e8910d
False                  vpc-12cf3b4f
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeVpcClassicLinkDnsSupport](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、リージョン eu-west-1 の VPC の ClassicLink DNS サポートステータスを記述します。

```
Get-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

出力:

```
ClassicLinkDnsSupported VpcId
-----
False                  vpc-0b12d3456a7e8910d
False                  vpc-12cf3b4f
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeVpcClassicLinkDnsSupport](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeVpcEndpointServices** を使用する

次のサンプルコードは、DescribeVpcEndpointServices を使用方法を説明しています。

CLI

AWS CLI

例 1: すべての VPC エンドポイントサービスを記述するには

次の describe-vpc-endpoint-services 例では、AWS リージョンのすべての VPC エンドポイントサービスを一覧表示します。

```
aws ec2 describe-vpc-endpoint-services
```

出力:

```
{
  "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"
]
}

```

例 2: エンドポイントサービスの詳細を記述するには

次の `describe-vpc-endpoint-services` の例では、Amazon S3 インターフェイスエンドポイントサービスの詳細を一覧表示します。

```
aws ec2 describe-vpc-endpoint-services \
  --filter 'Name=service-type,Values=Interface' Name=service-
name,Values=com.amazonaws.us-east-1.s3
```

出力:

```
{
  "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"
  ]
}
```

詳細については、AWS「PrivateLink ユーザーガイド」の「[使用可能な AWS サービス名](#)を表示する」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeVpcEndpointServices](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたフィルター (この場合は `com.amazonaws.eu-west-1.ecs`) を持つ EC2 VPC エンドポイントサービスについて記述します。さらに、`ServiceDetails` プロパティも展開され、詳細が表示されます。

```
Get-EC2VpcEndpointService -Region eu-west-1 -MaxResult 5 -Filter @{Name="service-name";Values="com.amazonaws.eu-west-1.ecs"} | Select-Object -ExpandProperty ServiceDetails
```

出力:

```
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
```

例 2: この例では、すべての EC2 VPC エンドポイントサービスを取得し、「`ssm`」に一致する `ServiceNames` を返します。

```
Get-EC2VpcEndpointService -Region eu-west-1 | Select-Object -ExpandProperty Servicenames | Where-Object { -match "ssm" }
```

出力:

```
com.amazonaws.eu-west-1.ssm
com.amazonaws.eu-west-1.ssmmessages
```


- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeVpcEndpointServices](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたフィルター (この場合は com.amazonaws.eu-west-1.ecs) を持つ EC2 VPC エンドポイントサービスについて記述します。さらに、ServiceDetails プロパティも展開され、詳細が表示されます。

```
Get-EC2VpcEndpointService -Region eu-west-1 -MaxResult 5 -Filter @{Name="service-name";Values="com.amazonaws.eu-west-1.ecs"} | Select-Object -ExpandProperty ServiceDetails
```

出力:

```
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
```

例 2: この例では、すべての EC2 VPC エンドポイントサービスを取得し、「ssm」に一致する ServiceNames を返します。

```
Get-EC2VpcEndpointService -Region eu-west-1 | Select-Object -ExpandProperty Servicenames | Where-Object { -match "ssm"}
```

出力:

```
com.amazonaws.eu-west-1.ssm
com.amazonaws.eu-west-1.ssmmessages
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeVpcEndpointServices](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeVpcEndpoints** を使用する

次のサンプルコードは、DescribeVpcEndpoints を使用する方法を説明しています。

CLI

AWS CLI

VPC エンドポイントを記述するには

次の describe-vpc-endpoints の例では、すべての VPC エンドポイントの詳細を表示します。

```
aws ec2 describe-vpc-endpoints
```

出力:

```
{
  "VpcEndpoints": [
    {
      "PolicyDocument": "{\"Version\":\"2008-10-17\",\"Statement\":\n[{\n\"Effect\":\"Allow\", \"Principal\": \"*\", \"Action\": \"*\", \"Resource\": \"*\n\"}]]\",
      "VpcId": "vpc-aabb1122",
      "NetworkInterfaceIds": [],
      "SubnetIds": [],
      "PrivateDnsEnabled": true,
      "State": "available",
      "ServiceName": "com.amazonaws.us-east-1.dynamodb",
      "RouteTableIds": [
        "rtb-3d560345"
      ],
      "Groups": [],
      "VpcEndpointId": "vpce-032a826a",
      "VpcEndpointType": "Gateway",
      "CreationTimestamp": "2017-09-05T20:41:28Z",
      "DnsEntries": [],
      "OwnerId": "123456789012"
    },
  ],
}
```

```

{
  "PolicyDocument": "{\n  \"Statement\": [\n    {\n      \"Action\":\n        \"*\", \n      \"Effect\": \"Allow\", \n      \"Principal\": \"*\", \n      \"Resource\": \"*\"\n    }\n  ],\n  \"VpcId\": \"vpc-1a2b3c4d\", \n  \"NetworkInterfaceIds\": [\n    \"eni-2ec2b084\", \n    \"eni-1b4a65cf\"\n  ],\n  \"SubnetIds\": [\n    \"subnet-d6fcaa8d\", \n    \"subnet-7b16de0c\"\n  ],\n  \"PrivateDnsEnabled\": false,\n  \"State\": \"available\", \n  \"ServiceName\": \"com.amazonaws.us-east-1.elasticloadbalancing\", \n  \"RouteTableIds\": [],\n  \"Groups\": [\n    {\n      \"GroupName\": \"default\", \n      \"GroupId\": \"sg-54e8bf31\"\n    }\n  ],\n  \"VpcEndpointId\": \"vpce-0f89a33420c1931d7\", \n  \"VpcEndpointType\": \"Interface\", \n  \"CreationTimestamp\": \"2017-09-05T17:55:27.583Z\", \n  \"DnsEntries\": [\n    {\n      \"HostedZoneId\": \"Z7HUB22UULQXV\", \n      \"DnsName\": \"vpce-0f89a33420c1931d7-bluzidnv.elasticloadbalancing.us-east-1.vpce.amazonaws.com\"\n    },\n    {\n      \"HostedZoneId\": \"Z7HUB22UULQXV\", \n      \"DnsName\": \"vpce-0f89a33420c1931d7-bluzidnv-us-east-1b.elasticloadbalancing.us-east-1.vpce.amazonaws.com\"\n    },\n    {\n      \"HostedZoneId\": \"Z7HUB22UULQXV\", \n      \"DnsName\": \"vpce-0f89a33420c1931d7-bluzidnv-us-east-1a.elasticloadbalancing.us-east-1.vpce.amazonaws.com\"\n    }\n  ],\n  \"OwnerId\": \"123456789012\"

```

```

    },
    {
      "VpcEndpointId": "vpce-aabbaabbaabbaabba",
      "VpcEndpointType": "GatewayLoadBalancer",
      "VpcId": "vpc-111122223333aabbcc",
      "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"
    }
  ]
}

```

詳細については、AWS PrivateLink ユーザーガイドの「[概念](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeVpcEndpoints](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、リージョン eu-west-1 の 1 つ以上の VPC エンドポイントについて記述します。次に、出力を次のコマンドにパイプします。このコマンドは VpcEndpointId プロパティを選択し、配列 VPC ID を文字列配列として返します。

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object -ExpandProperty
VpcEndpointId
```

出力:

```
vpce-01a2ab3f4f5cc6f7d
vpce-01d2b345a6787890b
```

```
vpce-0012e34d567890e12
vpce-0c123db4567890123
```

例 2: この例では、リージョン eu-west-1 のすべての vpc エンドポイントを記述し、VpcEndpointId、VpcId、ServiceName、および PrivateDnsEnabled プロパティを選択して表形式で表示します。

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object VpcEndpointId, VpcId,
ServiceName, PrivateDnsEnabled | Format-Table -AutoSize
```

出力:

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		

例 3: この例では、VPC エンドポイント vpce-01a2ab3f4f5cc6f7d のポリシードキュメントを JSON ファイルにエクスポートします。

```
Get-EC2VpcEndpoint -Region eu-west-1 -VpcEndpointId vpce-01a2ab3f4f5cc6f7d |
Select-Object -expand PolicyDocument | Out-File vpce_policyDocument.json
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeVpcEndpoints](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、リージョン eu-west-1 の 1 つ以上の VPC エンドポイントについて記述します。次に、出力を次のコマンドにパイプします。このコマンドは VpcEndpointId プロパティを選択し、配列 VPC ID を文字列配列として返します。

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object -ExpandProperty
VpcEndpointId
```

出力:

```
vpce-01a2ab3f4f5cc6f7d
vpce-01d2b345a6787890b
vpce-0012e34d567890e12
vpce-0c123db4567890123
```

例 2: この例では、リージョン eu-west-1 のすべての vpc エンドポイントを記述し、VpcEndpointId、VpcId、ServiceName、および PrivateDnsEnabled プロパティを選択して表形式で表示します。

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object VpcEndpointId, VpcId,
ServiceName, PrivateDnsEnabled | Format-Table -AutoSize
```

出力:

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		

例 3: この例では、VPC エンドポイント vpce-01a2ab3f4f5cc6f7d のポリシードキュメントを JSON ファイルにエクスポートします。

```
Get-EC2VpcEndpoint -Region eu-west-1 -VpcEndpointId vpce-01a2ab3f4f5cc6f7d |
Select-Object -expand PolicyDocument | Out-File vpce_policyDocument.json
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeVpcEndpoints](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DescribeVpcs**で を使用する

次のサンプルコードは、DescribeVpcs を使用方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [レジリエントなサービスの構築と管理](#)
- [Amazon VPC の使用を開始する](#)
- [Transit Gateway を開始する](#)
- [VPC IPAM の開始方法](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Get the default VPC for the account.
/// </summary>
/// <returns>The default VPC object.</returns>
public async Task<Vpc> GetDefaultVpc()
{
    try
    {
        var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
            new DescribeVpcsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new("is-default", new List<string>() { "true" })
                }
            }
        );
    }
}
```

```
        }
    });
    return vpcResponse.Vpcs[0];
}
catch (AmazonEC2Exception ec2Exception)
{
    if (ec2Exception.ErrorCode == "UnauthorizedOperation")
    {
        _logger.LogError(ec2Exception, $"You do not have the necessary
permissions to describe VPCs.");
    }

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

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[DescribeVpcs](#)」を参照してください。

CLI

AWS CLI

例 1: すべての VPC を説明するには

次の describe-vpcs の例では、VPC に関する詳細を取得します。

```
aws ec2 describe-vpcs
```

出力:

```
{
  "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"
            }
        ]
    }

```

```
    ]
  }
]
}
```

例 2: 指定した VPC を説明するには

次の `describe-vpcs` 例では、指定した VPC に関する詳細を取得します。

```
aws ec2 describe-vpcs \
  --vpc-ids vpc-06e4ab6c6cEXAMPLE
```

出力:

```
{
  "Vpcs": [
    {
      "CidrBlock": "10.0.0.0/16",
      "DhcpOptionsId": "dopt-19edf471",
      "State": "available",
      "VpcId": "vpc-06e4ab6c6cEXAMPLE",
      "OwnerId": "111122223333",
      "InstanceTenancy": "default",
      "CidrBlockAssociationSet": [
        {
          "AssociationId": "vpc-cidr-assoc-00b17b4eddEXAMPLE",
          "CidrBlock": "10.0.0.0/16",
          "CidrBlockState": {
            "State": "associated"
          }
        }
      ],
      "IsDefault": false,
      "Tags": [
        {
          "Key": "Name",
          "Value": "Shared VPC"
        }
      ]
    }
  ]
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeVpcs](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
const client = new EC2Client({});
const { Vpcs } = await client.send(
  new DescribeVpcsCommand({
    Filters: [{ Name: "is-default", Values: ["true"] }],
  }),
);
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[DescribeVpcs](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定した VPC を記述します。

```
Get-EC2Vpc -VpcId vpc-12345678
```

出力:

```
CidrBlock      : 10.0.0.0/16
DhcpOptionsId  : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault      : False
State          : available
```

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

例 2: この例では、デフォルトの VPC を記述します (リージョンごとに 1 つだけ指定できます)。アカウントがこのリージョンで EC2-Classic をサポートしている場合、デフォルトの VPC はありません。

```
Get-EC2Vpc -Filter @{Name="isDefault"; Values="true"}
```

出力:

```
CidrBlock      : 172.31.0.0/16  
DhcpOptionsId  : dopt-12345678  
InstanceTenancy : default  
IsDefault      : True  
State          : available  
Tags           : {}  
VpcId          : vpc-45678901
```

例 3: この例では、指定されたフィルターに一致する VPC を記述します (つまり、値「10.0.0.0/16」に一致する CIDR を持ち、状態が「available」である VPC)。

```
Get-EC2Vpc -Filter @{Name="cidr";  
  Values="10.0.0.0/16"},@{Name="state";Values="available"}
```

例 4: この例では、すべての VPC を記述します。

```
Get-EC2Vpc
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeVpcs](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定した VPC を記述します。

```
Get-EC2Vpc -VpcId vpc-12345678
```

出力:

```
CidrBlock      : 10.0.0.0/16
DhcpOptionsId  : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault      : False
State          : available
Tags           : {Name}
VpcId          : vpc-12345678
```

例 2: この例では、デフォルトの VPC を記述します (リージョンごとに 1 つだけ指定できます)。アカウントがこのリージョンで EC2-Classic をサポートしている場合、デフォルトの VPC はありません。

```
Get-EC2Vpc -Filter @{{Name="isDefault"; Values="true"}}
```

出力:

```
CidrBlock      : 172.31.0.0/16
DhcpOptionsId  : dopt-12345678
InstanceTenancy : default
IsDefault      : True
State          : available
Tags           : {}
VpcId          : vpc-45678901
```

例 3: この例では、指定されたフィルターに一致する VPC を記述します (つまり、値「10.0.0.0/16」に一致する CIDR を持ち、状態が「available」である VPC)。

```
Get-EC2Vpc -Filter @{{Name="cidr";
  Values="10.0.0.0/16"},@{{Name="state";Values="available"}}
```

例 4: この例では、すべての VPC を記述します。

```
Get-EC2Vpc
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeVpcs](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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."
            )
            log.error(
                "Ensure that your AWS IAM user or role has the correct"
                "permissions."
            )
        elif error_code == "InvalidParameterValue":
            log.error(
                "One or more parameters are invalid. Check the request"
                "parameters."
            )

```

```
        )

        log.error(f"Full error:\n\t{err}")
    else:
        if "Vpcs" in response and response["Vpcs"]:
            log.info(f"Retrieved default VPC: {response['Vpcs'][0]
['VpcId']}")
            return response["Vpcs"][0]
        else:
            pass
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[DescribeVpcs](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeVpnConnections** を使用する

次のサンプルコードは、DescribeVpnConnections を使用方法を説明しています。

CLI

AWS CLI

例 1: VPN 接続を記述するには

次の describe-vpn-connections の例では、すべての Site-to-Site VPN 接続について説明します。

```
aws ec2 describe-vpn-connections
```

出力:

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

```

詳細については、[AWS 「Site-to-Site VPN ユーザーガイド」](#) の Site-to-Site VPN works」を参照してください。

例 2: 使用可能な VPN 接続を記述するには

次の `describe-vpn-connections` の例では、状態が `available` である Site-to-Site VPN 接続について説明します。

```
aws ec2 describe-vpn-connections \
  --filters "Name=state,Values=available"
```

詳細については、[AWS 「Site-to-Site VPN ユーザーガイド」](#) の Site-to-Site VPN works」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeVpnConnections](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定された VPN 接続を記述します。

```
Get-EC2VpnConnection -VpnConnectionId vpn-12345678
```

出力:

```
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
```

例 2: この例では、保留中または使用可能な状態の VPN 接続を記述します。

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )
```

```
Get-EC2VpnConnection -Filter $filter
```

例 3: この例では、すべての VPN 接続を記述します。

```
Get-EC2VpnConnection
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeVpnConnections](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定された VPN 接続を記述します。

```
Get-EC2VpnConnection -VpnConnectionId vpn-12345678
```

出力:

```
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
```

例 2: この例では、保留中または使用可能な状態の VPN 接続を記述します。

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnConnection -Filter $filter
```

例 3: この例では、すべての VPN 接続を記述します。

```
Get-EC2VpnConnection
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeVpnConnections](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeVpnGateways** を使用する

次のサンプルコードは、DescribeVpnGateways を使用する方法を説明しています。

CLI

AWS CLI

仮想プライベートゲートウェイを記述するには

この例では、仮想プライベートゲートウェイを記述します。

コマンド:

```
aws ec2 describe-vpn-gateways
```

出力:

```
{
  "VpnGateways": [
    {
      "State": "available",
      "Type": "ipsec.1",
      "VpnGatewayId": "vgw-f211f09b",
      "VpcAttachments": [
        {
          "State": "attached",
          "VpcId": "vpc-98eb5ef5"
        }
      ]
    },
    {
      "State": "available",
      "Type": "ipsec.1",
      "VpnGatewayId": "vgw-9a4cacf3",
```

```

        "VpcAttachments": [
            {
                "State": "attaching",
                "VpcId": "vpc-a01106c2"
            }
        ]
    }
}

```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DescribeVpnGateways](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定された仮想プライベートゲートウェイを記述します。

```
Get-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

出力:

```

AvailabilityZone :
State            : available
Tags             : {}
Type             : ipsec.1
VpcAttachments   : {vpc-12345678}
VpnGatewayId     : vgw-1a2b3c4d

```

例 2: この例では、保留中または使用可能な状態の仮想プライベートゲートウェイを記述します。

```

$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnGateway -Filter $filter

```

例 3: この例では、仮想プライベートゲートウェイを記述します。

```
Get-EC2VpnGateway
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DescribeVpnGateways](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定された仮想プライベートゲートウェイを記述します。

```
Get-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

出力:

```
AvailabilityZone :  
State           : available  
Tags            : {}  
Type            : ipsec.1  
VpcAttachments  : {vpc-12345678}  
VpnGatewayId    : vgw-1a2b3c4d
```

例 2: この例では、保留中または使用可能な状態の仮想プライベートゲートウェイを記述します。

```
$filter = New-Object Amazon.EC2.Model.Filter  
$filter.Name = "state"  
$filter.Values = @( "pending", "available" )  
  
Get-EC2VpnGateway -Filter $filter
```

例 3: この例では、仮想プライベートゲートウェイを記述します。

```
Get-EC2VpnGateway
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DescribeVpnGateways](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DetachInternetGateway** を使用する

次のサンプルコードは、DetachInternetGateway を使用方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [プライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成する](#)
- [Amazon VPC の使用を開始する](#)

CLI

AWS CLI

VPC からインターネットゲートウェイをデタッチするには

次の detach-internet-gateway の例では、特定の VPC から指定したインターネットゲートウェイをデタッチします。

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

このコマンドでは何も出力されません。

詳細については、Amazon VPC ユーザーガイドの「[インターネットゲートウェイ](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DetachInternetGateway](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定された VPC から指定されたインターネットゲートウェイをデタッチします。

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DetachInternetGateway](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定された VPC から指定されたインターネットゲートウェイをデタッチします。

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の[DetachInternetGateway](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DetachNetworkInterface** を使用する

次のサンプルコードは、DetachNetworkInterface を使用方法を説明しています。

CLI

AWS CLI

インスタンスからネットワークインターフェイスをデタッチするには

この例では、指定されたインスタンスから指定されたネットワークインターフェイスをデタッチします。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 detach-network-interface --attachment-id eni-attach-66c4350a
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DetachNetworkInterface](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、ネットワークインターフェイスとインスタンス間の指定されたアタッチメントを削除します。

```
Dismount-EC2NetworkInterface -AttachmentId eni-attach-1a2b3c4d -Force
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DetachNetworkInterface](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、ネットワークインターフェイスとインスタンス間の指定されたアタッチメントを削除します。

```
Dismount-EC2NetworkInterface -AttachmentId eni-attach-1a2b3c4d -Force
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の[DetachNetworkInterface](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DetachVolume** を使用する

次のサンプルコードは、DetachVolume を使用する方法を説明しています。

CLI

AWS CLI

インスタンスからボリュームをデタッチするには

このコマンド例では、ボリューム (vol-049df61146c4d7901) をアタッチされているインスタンスからデタッチします。

コマンド:

```
aws ec2 detach-volume --volume-id vol-1234567890abcdef0
```

出力:

```
{
  "AttachTime": "2014-02-27T19:23:06.000Z",
  "InstanceId": "i-1234567890abcdef0",
  "VolumeId": "vol-049df61146c4d7901",
  "State": "detaching",
  "Device": "/dev/sdb"
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DetachVolume](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたボリュームをデタッチします。

```
Dismount-EC2Volume -VolumeId vol-12345678
```

出力:

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

例 2: インスタンス ID とデバイス名を指定して、正しいボリュームをデタッチするようにすることもできます。

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DetachVolume](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたボリュームをデタッチします。

```
Dismount-EC2Volume -VolumeId vol-12345678
```

出力:

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

例 2: インスタンス ID とデバイス名を指定して、正しいボリュームをデタッチするようにすることもできます。

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DetachVolume](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DetachVpnGateway** を使用する

次のサンプルコードは、DetachVpnGateway を使用する方法を説明しています。

CLI

AWS CLI

VPC から仮想プライベートゲートウェイをデタッチするには

この例では、指定された VPC から指定された仮想プライベートゲートウェイをデタッチします。コマンドが成功した場合、出力は返りません。

コマンド:

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

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DetachVpnGateway](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定された VPC から指定された仮想プライベートゲートウェイをデタッチします。

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DetachVpnGateway](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定された VPC から指定された仮想プライベートゲートウェイをデタッチします。

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DetachVpnGateway](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DisableVgwRoutePropagation** を使用する

次のサンプルコードは、`DisableVgwRoutePropagation` を使用方法を説明しています。

CLI

AWS CLI

ルート伝播を無効にするには

この例では、指定された仮想プライベートゲートウェイが指定されたルートテーブルに静的ルートを伝播しないようにします。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 disable-vgw-route-propagation --route-table-id rtb-22574640 --gateway-id vgw-9a4cacf3
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DisableVgwRoutePropagation](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、VGW が指定されたルーティングテーブルにルートを自動的に伝播することを無効にします。

```
Disable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DisableVgwRoutePropagation](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、VGW が指定されたルーティングテーブルにルートを自動的に伝播することを無効にします。

```
Disable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DisableVgwRoutePropagation](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DisableVpcClassicLink** を使用する

次のサンプルコードは、DisableVpcClassicLink を使用方法を説明しています。

CLI

AWS CLI

VPC の ClassicLink を無効にするには

この例では、vpc-88888888 の ClassicLink を無効にします。

コマンド:

```
aws ec2 disable-vpc-classic-link --vpc-id vpc-88888888
```

出力:

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

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DisableVpcClassicLink](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、vpc-01e23c4a5d6db78e9 の EC2VpcClassicLink を無効にします。True または False を返します。

```
Disable-EC2VpcClassicLink -VpcId vpc-01e23c4a5d6db78e9
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DisableVpcClassicLink](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、vpc-01e23c4a5d6db78e9 の EC2VpcClassicLink を無効にします。True または False を返します。

```
Disable-EC2VpcClassicLink -VpcId vpc-01e23c4a5d6db78e9
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DisableVpcClassicLink](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DisableVpcClassicLinkDnsSupport** を使用する

次のサンプルコードは、DisableVpcClassicLinkDnsSupport を使用方法を説明しています。

CLI

AWS CLI

VPC の ClassicLink DNS サポートを無効にするには

vpc-88888888 の ClassicLink DNS サポートを無効にします。

コマンド:

```
aws ec2 disable-vpc-classic-link-dns-support --vpc-id vpc-88888888
```

出力:

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

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DisableVpcClassicLinkDnsSupport](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、vpc-0b12d3456a7e8910d の ClassicLink DNS サポートを無効にします。

```
Disable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DisableVpcClassicLinkDnsSupport](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、vpc-0b12d3456a7e8910d の ClassicLink DNS サポートを無効にします。

```
Disable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DisableVpcClassicLinkDnsSupport](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DisassociateAddress**で 使用する

次のサンプルコードは、DisassociateAddress を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Disassociate an Elastic IP address from an EC2 instance.
/// </summary>
/// <param name="associationId">The association Id.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DisassociateIp(string associationId)
{
    try
```



```

    {
        var response = await _amazonEC2.DisassociateAddressAsync(
            new DisassociateAddressRequest { AssociationId =
associationId });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidAssociationID.NotFound")
        {
            _logger.LogError(
                $"AssociationId is invalid, unable to disassociate address.
{ec2Exception.Message}");
        }

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

```

- API の詳細については、AWS SDK for .NET API リファレンスの「[DisassociateAddress](#)」を参照してください。

Bash

AWS CLI Bash スクリプトを使用する

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
#####
```

```

# 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
}
```

この例で使用されているユーティリティ関数。

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

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DisassociateAddress](#)」を参照してください。

CLI

AWS CLI

EC2-Classic で Elastic IP アドレスの関連付けを解除するには

この例では、EC2-Classic のインスタンスから Elastic IP アドレスの関連付けを解除します。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 disassociate-address --public-ip 198.51.100.0
```

EC2-VPC で Elastic IP アドレスの関連付けを解除するには

この例では、VPC のインスタンスから Elastic IP アドレスの関連付けを解除します。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 disassociate-address --association-id eipassoc-2bebb745
```

- APIの詳細については、「AWS CLI Command Reference」の「[DisassociateAddress](#)」を参照してください。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

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

```
});  
  
    return response;  
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[DisassociateAddress](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import { DisassociateAddressCommand, EC2Client } from "@aws-sdk/client-ec2";  
  
/**  
 * Disassociate an Elastic IP address from an instance.  
 * @param {{ associationId: string }} options  
 */  
export const main = async ({ associationId }) => {  
    const client = new EC2Client({});  
    const command = new DisassociateAddressCommand({  
        // You can also use PublicIp, but that is for EC2 classic which is being  
        retired.  
        AssociationId: associationId,  
    });  
  
    try {  
        await client.send(command);  
        console.log("Successfully disassociated address");  
    } catch (caught) {  
        if (  
            caught instanceof Error &&  
            caught.name === "InvalidAssociationID.NotFound"  
        ) {
```

```
        console.warn(`${caught.message}.`);
    } else {
        throw caught;
    }
}
};
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[DisassociateAddress](#)」を参照してください。

Kotlin

SDK for Kotlin

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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!")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[DisassociateAddress](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、VPC 内の指定されたインスタンスから指定された Elastic IP アドレスの関連付けを解除します。

```
Unregister-EC2Address -AssociationId eipassoc-12345678
```

例 2: この例では、EC2-Classic 内の指定されたインスタンスから指定された Elastic IP アドレスの関連付けを解除します。

```
Unregister-EC2Address -PublicIp 203.0.113.17
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DisassociateAddress](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、VPC 内の指定されたインスタンスから指定された Elastic IP アドレスの関連付けを解除します。

```
Unregister-EC2Address -AssociationId eipassoc-12345678
```

例 2: この例では、EC2-Classic 内の指定されたインスタンスから指定された Elastic IP アドレスの関連付けを解除します。

```
Unregister-EC2Address -PublicIp 203.0.113.17
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DisassociateAddress](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。


```

class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
    actions using the client interface."""

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str]
            = None
        ) -> None:
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
            :param public_ip: The public IP address of the Elastic IP.
            :param instance_id: The ID of the associated EC2 instance, if any.
            """
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

        def __init__(self, ec2_client: Any) -> None:
            """
            Initializes the ElasticIpWrapper with an EC2 client.

            :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
            level
                                access to AWS EC2 services.
            """
            self.ec2_client = ec2_client
            self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
        client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

```

```

def disassociate(self, allocation_id: str) -> None:
    """
    Removes an association between an Elastic IP address and an instance.
    When the
    association is removed, the instance is assigned a new public IP address.

    :param allocation_id: The allocation ID of the Elastic IP to
    disassociate.
    :raises ClientError: If the disassociation fails, such as when the
    association ID is not found.
    """
    elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
    allocation_id)
    if elastic_ip is None or elastic_ip.instance_id is None:
        logger.info(
            f"No association found for Elastic IP with allocation ID
    {allocation_id}."
        )
        return

    try:
        # Retrieve the association ID before disassociating
        response =
    self.ec2_client.describe_addresses(AllocationIds=[allocation_id])
        association_id = response["Addresses"][0].get("AssociationId")

        if association_id:

    self.ec2_client.disassociate_address(AssociationId=association_id)
        elastic_ip.instance_id = None # Remove the instance association
    else:
        logger.info(
            f"No Association ID found for Elastic IP with allocation ID
    {allocation_id}."
        )

    except ClientError as err:
        if err.response["Error"]["Code"] == "InvalidAssociationID.NotFound":
            logger.error(
                f"Failed to disassociate Elastic IP {allocation_id} "
                "because the specified association ID for the Elastic IP
    address was not found. "

```

```

        "Verify the association ID and ensure the Elastic IP is
        currently associated with a "
        "resource before attempting to disassociate it."
    )
    raise

```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[DisassociateAddress](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

pub async fn disassociate_ip_address(&self, association_id: &str) ->
Result<(), EC2Error> {
    self.client
        .disassociate_address()
        .association_id(association_id)
        .send()
        .await?;
    Ok(())
}

```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[DisassociateAddress](#)」を参照してください。

Swift

SDK for Swift

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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)")
    }
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[DisassociateAddress](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DisassociateRouteTable** を使用する

次のサンプルコードは、DisassociateRouteTable を使用方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [Amazon VPC の使用を開始する](#)

CLI

AWS CLI

ルートテーブルの関連付けを解除するには

この例では、指定されたサブネットから指定されたルートテーブルの関連付けを解除します。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 disassociate-route-table --association-id rtbassoc-781d0d1a
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[DisassociateRouteTable](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、ルートテーブルとサブネット間の指定された関連付けを削除します。

```
Unregister-EC2RouteTable -AssociationId rtbassoc-1a2b3c4d
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DisassociateRouteTable](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、ルートテーブルとサブネット間の指定された関連付けを削除します。

```
Unregister-EC2RouteTable -AssociationId rtbassoc-1a2b3c4d
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DisassociateRouteTable](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **EnableVgwRoutePropagation** を使用する

次のサンプルコードは、EnableVgwRoutePropagation を使用する方法を説明しています。

CLI

AWS CLI

ルート伝播を有効にするには

この例では、指定された仮想プライベートゲートウェイが指定されたルートテーブルに静的ルートを伝播できるようにします。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 enable-vgw-route-propagation --route-table-id rtb-22574640 --gateway-id vgw-9a4cacf3
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[EnableVgwRoutePropagation](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定された VGW が指定されたルーティングテーブルに自動的にルートを伝播できるようにします。

```
Enable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[EnableVgwRoutePropagation](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定された VGW が指定されたルーティングテーブルに自動的にルートを伝播できるようにします。

```
Enable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[EnableVgwRoutePropagation](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **EnableVolumeIo** を使用する

次のサンプルコードは、EnableVolumeIo を使用方法を説明しています。

CLI

AWS CLI

ボリュームの I/O を有効にするには

この例では、ボリューム vol-1234567890abcdef0 の I/O を有効にします。

コマンド:

```
aws ec2 enable-volume-io --volume-id vol-1234567890abcdef0
```

出力:

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

- API の詳細については、「AWS CLI コマンドリファレンス」の「[EnableVolumeIo](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、I/O オペレーションが無効になっている場合に、指定されたボリュームの I/O オペレーションを有効にします。

```
Enable-EC2VolumeIO -VolumeId vol-12345678
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[EnableVolumelo](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、I/O オペレーションが無効になっている場合に、指定されたボリュームの I/O オペレーションを有効にします。

```
Enable-EC2VolumeIO -VolumeId vol-12345678
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[EnableVolumelo](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **EnableVpcClassicLink** を使用する

次のサンプルコードは、EnableVpcClassicLink を使用方法を説明しています。

CLI

AWS CLI

ClassicLink 用に VPC を有効にするには

この例では、ClassicLink の vpc-88888888 を有効にします。

コマンド:

```
aws ec2 enable-vpc-classic-link --vpc-id vpc-88888888
```

出力:

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


- API の詳細については、「AWS CLI コマンドリファレンス」の「[EnableVpcClassicLink](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、ClassicLink の VPC vpc-0123456b789b0d12f を有効にします。

```
Enable-EC2VpcClassicLink -VpcId vpc-0123456b789b0d12f
```

出力:

```
True
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[EnableVpcClassicLink](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、ClassicLink の VPC vpc-0123456b789b0d12f を有効にします。

```
Enable-EC2VpcClassicLink -VpcId vpc-0123456b789b0d12f
```

出力:

```
True
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[EnableVpcClassicLink](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **EnableVpcClassicLinkDnsSupport** を使用する

次のサンプルコードは、EnableVpcClassicLinkDnsSupport を使用する方法を説明しています。

CLI

AWS CLI

VPC の ClassicLink DNS サポートを有効にするには

この例では、vpc-88888888 の ClassicLink DNS サポートを有効にします。

コマンド:

```
aws ec2 enable-vpc-classic-link-dns-support --vpc-id vpc-88888888
```

出力:

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

- API の詳細については、「AWS CLI コマンドリファレンス」の「[EnableVpcClassicLinkDnsSupport](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、vpc-0b12d3456a7e8910d が ClassicLink の DNS ホスト名解決をサポートできるようにします。

```
Enable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[EnableVpcClassicLinkDnsSupport](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、vpc-0b12d3456a7e8910d が ClassicLink の DNS ホスト名解決をサポートできるようにします。

```
Enable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[EnableVpcClassicLinkDnsSupport](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で `GetConsoleOutput` を使用する

次のサンプルコードは、`GetConsoleOutput` を使用する方法を説明しています。

CLI

AWS CLI

例 1: コンソール出力を取得するには

次の `get-console-output` の例では、指定した Linux インスタンスのコンソール出力を取得します。

```
aws ec2 get-console-output \  
  --instance-id i-1234567890abcdef0
```

出力:

```
{  
  "InstanceId": "i-1234567890abcdef0",  
  "Timestamp": "2013-07-25T21:23:53.000Z",  
  "Output": "..."  
}
```

詳細については、「Amazon EC2 ユーザーガイド」の「[インスタンスコンソール出力](#)」を参照してください。

例 2: 最新のコンソール出力を取得するには

次の `get-console-output` の例では、指定した Linux インスタンスの最新のコンソール出力を取得します。

```
aws ec2 get-console-output \  
  --instance-id i-1234567890abcdef0 \  
  --latest \  
  --output text
```

出力:

```
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
```

詳細については、「Amazon EC2 ユーザーガイド」の「[インスタンスコンソール出力](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GetConsoleOutput](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定した Linux インスタンスのコンソール出力を取得します。コンソール出力はエンコードされます。

```
Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456
```

出力:

InstanceId	Output
i-0e194d3c47c123637	WyAgICAwLjAwMDAwMF0gQ29tbW...bGU9dHR5UzAgc2Vs

例 2: この例では、エンコードされたコンソール出力を変数に保存してからデコードします。

```
$Output_encoded = (Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456).Output
[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($Output_encoded))
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[GetConsoleOutput](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定した Linux インスタンスのコンソール出力を取得します。コンソール出力はエンコードされます。

```
Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456
```

出力:

InstanceId	Output
-----	-----
i-0e194d3c47c123637	WyAgICAwLjAwMDAwMF0gQ29tbW...bGU9dHR5UzAgc2Vs

例 2: この例では、エンコードされたコンソール出力を変数に保存してからデコードします。

```
$Output_encoded = (Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456).Output  
[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($Output_encoded))
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[GetConsoleOutput](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **GetHostReservationPurchasePreview** を使用する

次のサンプルコードは、GetHostReservationPurchasePreview を使用する方法を説明しています。

CLI

AWS CLI

専有ホストの予約の購入プレビューを取得するには

この例では、アカウント内の指定された専有ホストに対して指定された専有ホスト予約のコストをプレビューします。

コマンド:

```
aws ec2 get-host-reservation-purchase-preview --offering-id hro-03f707bf363b6b324
--host-id-set h-013abcd2a00cbd123
```

出力:

```
{
  "TotalHourlyPrice": "1.499",
  "Purchase": [
    {
      "HourlyPrice": "1.499",
      "InstanceFamily": "m4",
      "PaymentOption": "NoUpfront",
      "HostIdSet": [
        "h-013abcd2a00cbd123"
      ],
      "UpfrontPrice": "0.000",
      "Duration": 31536000
    }
  ],
  "TotalUpfrontPrice": "0.000"
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[GetHostReservationPurchasePreview](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、専有ホスト h-01e23f4cd567890f1 の構成と一致する設定の予約購入をレビューします。

```
Get-EC2HostReservationPurchasePreview -OfferingId hro-0c1f23456789d0ab -HostIdSet
h-01e23f4cd567890f1
```

出力:

CurrencyCode	Purchase	TotalHourlyPrice	TotalUpfrontPrice
-----	-----	-----	-----
	{}	1.307	0.000

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[GetHostReservationPurchasePreview](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、専有ホスト h-01e23f4cd567890f1 の構成と一致する設定の予約購入をプレビューします。

```
Get-EC2HostReservationPurchasePreview -OfferingId hro-0c1f23456789d0ab -HostIdSet h-01e23f4cd567890f1
```

出力:

CurrencyCode	Purchase	TotalHourlyPrice	TotalUpfrontPrice
-----	-----	-----	-----
	{}	1.307	0.000

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[GetHostReservationPurchasePreview](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **GetPasswordData**で を使用する

次のサンプルコードは、GetPasswordData を使用する方法を説明しています。

CLI

AWS CLI

暗号化されたパスワードを取得するには

この例では、暗号化されたパスワードを取得します。

コマンド:

```
aws ec2 get-password-data --instance-id i-1234567890abcdef0
```

出力:

```
{
  "InstanceId": "i-1234567890abcdef0",
  "Timestamp": "2013-08-07T22:18:38.000Z",
  "PasswordData": "gSlJFq+VpcZXqy+iktxMF6NyxQ4qCrT4+gaOuN0enX1MmgXPTj7XEXAMPLE
UQ+YeFfb+L1U4C4AKv652Ux1iRB3CPTYP7WmU3TUnhsuBd+p6LVk7T2lKUm160Xbk6WPW1VYYm/TRPB1
e1DQ7PY4an/DgZT4mwcpRFIGzhniQgDDe01InvSDcwoUTwNs0Y1S8ouri2W4n5GNlriM3Q0AnNVelVz/
53TkDtxbNoU606M1gK9zUWSxqEgwvbV2j8c5rP0WCuaMWSF14ziDu4bd7q+4RSyi8NUsVWnKZ4aEZffu
DPGzKrF5yL1f3etP2L4ZR6CvG7K1hx7VK0QVN32Dajw=="
}
```

復号されたパスワードを取得するには

この例では、復号されたパスワードを取得します。

コマンド:

```
aws ec2 get-password-data --instance-id i-1234567890abcdef0 --priv-launch-key C:
\Keys\MyKeyPair.pem
```

出力:

```
{
  "InstanceId": "i-1234567890abcdef0",
  "Timestamp": "2013-08-30T23:18:05.000Z",
  "PasswordData": "&ViJ652e*u"
}
```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[GetPasswordData](#)」を参照してください。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。


```
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);
}
}

```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[GetPasswordData](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、Amazon EC2 が指定された Windows インスタンスの管理者アカウントに割り当てたパスワードを復号します。pem ファイルを指定すると、-Decrypt スイッチの設定が自動的に適用されます。

```
Get-EC2PasswordData -InstanceId i-12345678 -PemFile C:\path\my-key-pair.pem
```

出力:

```
mYZ(PA9?C)Q
```

例 2: (Windows PowerShell のみ) インスタンスを検査して、インスタンスの起動に使用されるキーペアの名前を判断し、AWS Toolkit for Visual Studio の設定ストアで対応するキーペアデータを見つけようとします。キーペアデータが見つかった場合、パスワードは復号されます。

```
Get-EC2PasswordData -InstanceId i-12345678 -Decrypt
```

出力:

```
mYZ(PA9?C)Q
```

例 3: インスタンスの暗号化されたパスワードデータを返します。

```
Get-EC2PasswordData -InstanceId i-12345678
```

出力:

```
iVz3BAK/WAXV.....dqt8WeMA==
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[GetPasswordData](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、Amazon EC2 が指定された Windows インスタンスの管理者アカウントに割り当てたパスワードを復号します。pem ファイルを指定すると、-Decrypt スイッチの設定が自動的に適用されます。

```
Get-EC2PasswordData -InstanceId i-12345678 -PemFile C:\path\my-key-pair.pem
```

出力:

```
mYZ(PA9?C)Q
```

例 2: (Windows PowerShell のみ) インスタンスを検査して、インスタンスの起動に使用されるキーペアの名前を判断し、AWS Toolkit for Visual Studio の設定ストアで対応するキーペアデータを見つけようとします。キーペアデータが見つかった場合、パスワードは復号されます。

```
Get-EC2PasswordData -InstanceId i-12345678 -Decrypt
```

出力:

```
mYZ(PA9?C)Q
```

例 3: インスタンスの暗号化されたパスワードデータを返します。

```
Get-EC2PasswordData -InstanceId i-12345678
```

出力:

```
iVz3BAK/WAXV.....dqt8WeMA==
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[GetPasswordData](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **ImportImage** を使用する

次のサンプルコードは、ImportImage を使用する方法を説明しています。

CLI

AWS CLI

VM イメージファイルを AMI としてインポートするには

次の import-image の例では、指定した OVA をインポートします。

```
aws ec2 import-image \  
  --disk-containers Format=ova,UserBucket="{S3Bucket=my-import-bucket,S3Key=vms/  
my-server-vm.ova}"
```

出力:

```
{  
  "ImportTaskId": "import-ami-1234567890abcdef0",  
  "Progress": "2",  
  "SnapshotDetails": [  
    {  
      "DiskImageSize": 0.0,  
      "Format": "ova",  
      "UserBucket": {  
        "S3Bucket": "my-import-bucket",  
        "S3Key": "vms/my-server-vm.ova"  
      }  
    }  
  ],  
  "Status": "active",  
  "StatusMessage": "pending"  
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ImportImage](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、べき等性トークンを使用して、指定された Amazon S3 バケットから Amazon EC2 にシングルディスク仮想マシンイメージをインポートします。この例では、「VM Import Prerequisites」トピックで説明されているように、デフォルトの名前が「vmimport」の VM Import サービスロールが存在し、指定されたバケットへの Amazon EC2 アクセスを許可するポリシーが付与されている必要があります。カスタムロールを使用するには、**-RoleName** パラメータを使用してロール名を指定します。

```
$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
```

出力:

```
Architecture      :
Description       : Windows 2008 Standard Image
Hypervisor        :
ImageId           :
ImportTaskId      : import-ami-abcdefgh
LicenseType       : AWS
Platform          : Windows
Progress          : 2
SnapshotDetails   : {}
Status            : active
StatusMessage     : pending
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ImportImage](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、べき等性トークンを使用して、指定された Amazon S3 バケットから Amazon EC2 にシングルディスク仮想マシンイメージをインポートします。この例では、「VM Import Prerequisites」トピックで説明されているように、デフォルトの名前が「vmimport」の VM Import サービスロールが存在し、指定されたバケットへの Amazon EC2 アクセスを許可するポリシーが付与されている必要があります。カスタムロールを使用するには、**-RoleName** パラメータを使用してロール名を指定します。

```
$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
```

出力:

```
Architecture      :
Description       : Windows 2008 Standard Image
Hypervisor        :
ImageId           :
ImportTaskId      : import-ami-abcdefgh
LicenseType       : AWS
Platform          : Windows
Progress          : 2
SnapshotDetails   : {}
Status            : active
StatusMessage     : pending
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[ImportImage](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **ImportKeyPair** を使用する

次のサンプルコードは、`ImportKeyPair` を使用する方法を説明しています。

CLI

AWS CLI

パブリックキーをインポートするには

最初に、選択したツールを使用してキーペアを生成します。例えば、次の `ssh-keygen` コマンドを使用します。

コマンド:

```
ssh-keygen -t rsa -C "my-key" -f ~/.ssh/my-key
```

出力:

```
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.  
...
```

このコマンド例では、指定されたパブリックキーをインポートします。

コマンド:

```
aws ec2 import-key-pair --key-name "my-key" --public-key-material fileb://~/.ssh/  
my-key.pub
```

出力:

```
{
```



```
"KeyName": "my-key",
"KeyFingerprint": "1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca"
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ImportKeyPair](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、パブリックキーを EC2 にインポートします。最初の行は、パブリックキーファイル (*.pub) の内容を変数 **\$publickey** に保存します。次に、パブリックキーファイルの UTF8 形式を Base64 でエンコードされた文字列に変換し、変換された文字列を変数 **\$pkbase64** に保存します。最後の行では、変換されたパブリックキーが EC2 にインポートされます。コマンドレットは、キーフィンガープリントと名前を結果として返します。

```
$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
```

出力:

KeyFingerprint	KeyName
-----	-----
do:d0:15:8f:79:97:12:be:00:fd:df:31:z3:b1:42:z1	Example-user-key

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ImportKeyPair](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、パブリックキーを EC2 にインポートします。最初の行は、パブリックキーファイル (*.pub) の内容を変数 **\$publickey** に保存します。次に、パブリックキーファイルの UTF8 形式を Base64 でエンコードされた文字列に変換し、変換された文字列を変数 **\$pkbase64** に保存します。最後の行では、変換されたパブリックキーが EC2 にインポートされます。コマンドレットは、キーフィンガープリントと名前を結果として返します。

```
$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
```

出力:

```
KeyFingerprint                                KeyName
-----
do:d0:15:8f:79:97:12:be:00:fd:df:31:z3:b1:42:z1 Example-user-key
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[ImportKeyPair](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **ImportSnapshot** を使用する

次のサンプルコードは、ImportSnapshot を使用方法を説明しています。

CLI

AWS CLI

スナップショットをインポートするには

次の import-snapshot の例では、指定したディスクをスナップショットとしてインポートします。

```
aws ec2 import-snapshot \
  --description "My server VMDK" \
  --disk-container Format=VMDK,UserBucket={'S3Bucket=my-import-
bucket,S3Key=vms/my-server-vm.vmdk'}
```

出力:

```
{
  "Description": "My server VMDK",
```

```
"ImportTaskId": "import-snap-1234567890abcdef0",
"SnapshotTaskDetail": {
  "Description": "My server VMDK",
  "DiskImageSize": "0.0",
  "Format": "VMDK",
  "Progress": "3",
  "Status": "active",
  "StatusMessage": "pending"
  "UserBucket": {
    "S3Bucket": "my-import-bucket",
    "S3Key": "vms/my-server-vm.vmdk"
  }
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ImportSnapshot](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、「VMDK」形式の VM ディスクイメージを Amazon EBS スナップショットにインポートします。この例では、<http://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/VMImportPrerequisites.html> の **VM Import Prerequisites** トピックで説明されているように、デフォルトの名前が「vmimport」の VM Import サービスロールと、指定されたバケットへの Amazon EC2 アクセスを許可するポリシーが必要です。カスタムロールを使用するには、**-RoleName** パラメータを使用してロール名を指定します。

```
$parms = @{
  "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 @parms
```

出力:

Description	ImportTaskId	SnapshotTaskDetail
-----	-----	-----
Disk Image Import	import-snap-abcdefgh	Amazon.EC2.Model.SnapshotTaskDetail

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ImportSnapshot](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、「VMDK」形式の VM ディスクイメージを Amazon EBS スナップショットにインポートします。この例では、<http://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/VMImportPrerequisites.html> の **VM Import Prerequisites** トピックで説明されているように、デフォルトの名前が「vmimport」の VM Import サービスロールと、指定されたバケットへの Amazon EC2 アクセスを許可するポリシーが必要です。カスタムロールを使用するには、**-RoleName** パラメータを使用してロール名を指定します。

```
$parms = @{
    "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 @parms
```

出力:

Description	ImportTaskId	SnapshotTaskDetail
-----	-----	-----
Disk Image Import	import-snap-abcdefgh	Amazon.EC2.Model.SnapshotTaskDetail

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[ImportSnapshot](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で `ModifyCapacityReservation` を使用する

次のサンプルコードは、`ModifyCapacityReservation` を使用する方法を説明しています。

CLI

AWS CLI

例 1: 既存のキャパシティ予約で予約されたインスタンスの数を変更するには

次の `modify-capacity-reservation` の例では、キャパシティ予約でキャパシティを予約するインスタンスの合計数を変更します。

```
aws ec2 modify-capacity-reservation \  
  --capacity-reservation-id cr-1234abcd56EXAMPLE \  
  --instance-count 5
```

出力:

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

詳細については、Amazon EC2 ユーザーガイドの「[キャパシティ予約の変更](#)」を参照してください。

例 2: 既存のキャパシティ予約の終了日時を変更するには

次の `modify-capacity-reservation` の例では、指定した日時に既存のキャパシティ予約が終了するように変更します。

```
aws ec2 modify-capacity-reservation \  
  --capacity-reservation-id cr-1234abcd56EXAMPLE \  
  --end-date-type limited \  
  --end-date 2019-08-31T23:59:59Z
```

詳細については、Amazon EC2 ユーザーガイドの「[キャパシティ予約の変更](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ModifyCapacityReservation](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、インスタンス数を 1 に変更して CapacityReservationId cr-0c1f2345db6f7cdba を変更します。

```
Edit-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba -  
InstanceCount 1
```

出力:

```
True
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ModifyCapacityReservation](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、インスタンス数を 1 に変更して CapacityReservationId cr-0c1f2345db6f7cdba を変更します。

```
Edit-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba -  
InstanceCount 1
```

出力:

```
True
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[ModifyCapacityReservation](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **ModifyHosts** を使用する

次のサンプルコードは、ModifyHosts を使用方法を説明しています。

CLI

AWS CLI

例 1: 専有ホストの自動配置を有効にするには

次の modify-hosts の例では、インスタンスタイプ設定が一致するすべてのターゲット未指定のインスタンス起動を受け入れるため、専有ホストの自動配置を有効にします。

```
aws ec2 modify-hosts \  
  --host-id h-06c2f189b4EXAMPLE \  
  --auto-placement on
```

出力:

```
{  
  "Successful": [  
    "h-06c2f189b4EXAMPLE"  
  ],  
  "Unsuccessful": []  
}
```

詳細については、Amazon EC2 ユーザーガイドの「[専有ホストの自動配置設定の変更](#)」を参照してください。

例 2: 専有ホストのホストリカバリを有効にするには

次の modify-hosts の例では、指定した専有ホストのホストの復旧を有効にします。

```
aws ec2 modify-hosts \  
  --host-id h-06c2f189b4EXAMPLE \  
  --host-recovery on
```

出力:

```
{  
  "Successful": [  
    "h-06c2f189b4EXAMPLE"  
  ]  
}
```

```
    ],
    "Unsuccessful": []
}
```

詳細については、Amazon EC2 ユーザーガイドの「[専有ホストの自動配置設定の変更](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ModifyHosts](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、専有ホスト h-01e23f4cd567890f3 の 自動配置設定をオフに変更します。

```
Edit-EC2Host -HostId h-03e09f8cd681609f3 -AutoPlacement off
```

出力:

```
Successful          Unsuccessful
-----
{h-01e23f4cd567890f3} {}
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ModifyHosts](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、専有ホスト h-01e23f4cd567890f3 の 自動配置設定をオフに変更します。

```
Edit-EC2Host -HostId h-03e09f8cd681609f3 -AutoPlacement off
```

出力:

```
Successful          Unsuccessful
-----
{h-01e23f4cd567890f3} {}
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[ModifyHosts](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **ModifyIdFormat** を使用する

次のサンプルコードは、ModifyIdFormat を使用方法を説明しています。

CLI

AWS CLI

リソースの長い ID 形式を有効にするには

次の modify-id-format の例では、instance リソースタイプの長い ID 形式を有効にします。

```
aws ec2 modify-id-format \  
  --resource instance \  
  --use-long-ids
```

リソースの長い ID 形式を無効にするには

次の modify-id-format の例では、instance リソースタイプの長い ID 形式を無効にします。

```
aws ec2 modify-id-format \  
  --resource instance \  
  --no-use-long-ids
```

次の modify-id-format の例では、オプトイン期間内にあるすべてのサポート対象リソースタイプの長い ID 形式を有効にします。

```
aws ec2 modify-id-format \  
  --resource all-current \  
  --use-long-ids
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ModifyIdFormat](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたリソースタイプの長い ID 形式を有効にします。

```
Edit-EC2IdFormat -Resource instance -UseLongId $true
```

例 2: この例では、指定されたリソースタイプの長い ID 形式を無効にします。

```
Edit-EC2IdFormat -Resource instance -UseLongId $false
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ModifyIdFormat](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたリソースタイプの長い ID 形式を有効にします。

```
Edit-EC2IdFormat -Resource instance -UseLongId $true
```

例 2: この例では、指定されたリソースタイプの長い ID 形式を無効にします。

```
Edit-EC2IdFormat -Resource instance -UseLongId $false
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[ModifyIdFormat](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください [AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **ModifyImageAttribute** を使用する

次のサンプルコードは、ModifyImageAttribute を使用する方法を説明しています。

CLI

AWS CLI

例 1: AMI をパブリックにするには

次の `modify-instance-attribute` の例では、指定した AMI を公開します。

```
aws ec2 modify-image-attribute \  
  --image-id ami-5731123e \  
  --launch-permission "Add=[{Group=all}]"
```

このコマンドでは何も出力されません。

例 2: AMI をプライベートにするには

次の `modify-instance-attribute` の例では、指定した AMI を非公開にします。

```
aws ec2 modify-image-attribute \  
  --image-id ami-5731123e \  
  --launch-permission "Remove=[{Group=all}]"
```

このコマンドでは何も出力されません。

例 3: AWS アカウントに起動アクセス許可を付与するには

次の `modify-instance-attribute` 例では、指定された AWS アカウントに起動アクセス許可を付与します。

```
aws ec2 modify-image-attribute \  
  --image-id ami-5731123e \  
  --launch-permission "Add=[{UserId=123456789012}]"
```

このコマンドでは何も出力されません。

例 4: AWS アカウントから起動許可を削除するには

次の `modify-instance-attribute` 例では、指定された AWS アカウントから起動許可を削除します。

```
aws ec2 modify-image-attribute \  
  --image-id ami-5731123e \  
  --launch-permission "Remove=[{UserId=123456789012}]"
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ModifyImageAttribute](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定された AMI の説明を更新します。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Description "New description"
```

例 2: この例では、AMI を公開します (例: 誰でも AWS アカウント 使用できる)。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType add -UserGroup all
```

例 3: この例では、AMI を非公開にします (例えば、所有者である自分だけが使用できるようにします)。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType remove -UserGroup all
```

例 4: この例では、指定された に起動アクセス許可を付与します AWS アカウント。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType add -UserId 111122223333
```

例 5: この例では、指定された から起動アクセス許可を削除します AWS アカウント。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType remove -UserId 111122223333
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ModifyImageAttribute](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定された AMI の説明を更新します。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Description "New description"
```

例 2: この例では、AMI を公開します (たとえば、誰でも使用 AWS アカウント できるようにします)。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType add -UserGroup all
```

例 3: この例では、AMI を非公開にします (例えば、所有者である自分だけが使用できるようにします)。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType remove -UserGroup all
```

例 4: この例では、指定された に起動アクセス許可を付与します AWS アカウント。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType add -UserId 111122223333
```

例 5: この例では、指定された から起動アクセス許可を削除します AWS アカウント。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType remove -UserId 111122223333
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の [ModifyImageAttribute](#) を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください [AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **ModifyInstanceAttribute** を使用する

次のサンプルコードは、ModifyInstanceAttribute を使用する方法を説明しています。

CLI

AWS CLI

例 1: インスタンスタイプを変更するには

次の modify-instance-attribute の例では、指定したインスタンスのインスタンスタイプを変更します。インスタンスは stopped の状態である必要があります。

```
aws ec2 modify-instance-attribute \  
  --instance-id i-1234567890abcdef0 \  
  --instance-type "{\"Value\": \"m1.small\"}"
```

このコマンドでは何も出力されません。

例 2: インスタンスで拡張ネットワーキングを有効にするには

次の modify-instance-attribute の例では、指定されたインスタンスの拡張ネットワーキングを有効にします。インスタンスは stopped の状態である必要があります。

```
aws ec2 modify-instance-attribute \  
  --instance-id i-1234567890abcdef0 \  
  --sriov-net-support simple
```

このコマンドでは何も出力されません。

例 3: sourceDestCheck 属性を変更するには

次の modify-instance-attribute の例では、指定したインスタンスの sourceDestCheck 属性を true に設定します。インスタンスは VPC にある必要があります。

```
aws ec2 modify-instance-attribute --instance-id i-1234567890abcdef0 --source-  
dest-check "{\"Value\": true}"
```

このコマンドでは何も出力されません。

例 4: ルートボリュームの deleteOnTermination 属性を変更するには

次の modify-instance-attribute の例では、指定した Amazon EBS-backed インスタンスのルートボリュームに対する deleteOnTermination 属性を false に設定します。デフォルトでは、この属性はルートボリュームに対して true です。

コマンド:

```
aws ec2 modify-instance-attribute \  
  --instance-id i-1234567890abcdef0 \  
  --block-device-mappings "[{\"DeviceName\": \"/dev/sda1\", \"Ebs\":  
{\"DeleteOnTermination\": false}}]"
```

このコマンドでは何も出力されません。

例 5: インスタンスにアタッチされたユーザーデータを変更するには

次の `modify-instance-attribute` の例では、指定したインスタンスの `UserData` としてファイル `UserData.txt` の内容を追加します。

元のファイル `UserData.txt` の内容:

```
#!/bin/bash
yum update -y
service httpd start
chkconfig httpd on
```

ファイルの内容は、base64 でエンコードされている必要があります。最初のコマンドはテキストファイルを base64 に変換し、新しいファイルとして保存します。

コマンドの Linux/macOS バージョン:

```
base64 UserData.txt > UserData.base64.txt
```

このコマンドでは何も出力されません。

コマンドの Windows バージョン:

```
certutil -encode UserData.txt tmp.b64 && findstr /v /c:- tmp.b64 >
UserData.base64.txt
```

出力:

```
Input Length = 67
Output Length = 152
CertUtil: -encode command completed successfully.
```

これで、次の CLI コマンドでそのファイルを参照できるようになりました。

```
aws ec2 modify-instance-attribute \
  --instance-id=i-09b5a14dbca622e76 \
  --attribute userData --value file:///UserData.base64.txt
```

このコマンドでは何も出力されません。

詳細については、EC2 ユーザーガイドの「[ユーザーデータと AWS CLI](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ModifyInstanceAttribute](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたインスタンスのインスタンスタイプを変更します。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceType m3.medium
```

例 2: この例では、単ルート I/O 仮想化 (SR-IOV) ネットワークサポートパラメータ - SrvNetSupport の値に「simple」を指定して、指定されたインスタンスの拡張ネットワーキングを有効にします。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SrvNetSupport "simple"
```

例 3: この例では、指定されたインスタンスのセキュリティグループを変更します。インスタンスは VPC にある必要があります。名前ではなく、各セキュリティグループの ID を指定する必要があります。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -Group @( "sg-12345678",  
"sg-45678901" )
```

例 4: この例では、指定されたインスタンスの EBS I/O 最適化を有効にします。この機能は、すべてのインスタンスタイプで利用できるわけではありません。EBS 最適化インスタンスを使用する場合は、追加料金が適用されます。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -EbsOptimized $true
```

例 5: この例では、指定されたインスタンスの送信元/送信先チェックを有効にします。NAT インスタンスが NAT を実行するには、値が「false」である必要があります。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SourceDestCheck $true
```

例 6: この例では、指定されたインスタンスの終了を無効にします。


```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -DisableApiTermination $true
```

例 7: この例では、インスタンスからシャットダウンを開始したときに終了するように、指定されたインスタンスを変更します。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -  
InstanceInitiatedShutdownBehavior terminate
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ModifyInstanceAttribute](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたインスタンスのインスタンスタイプを変更します。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceType m3.medium
```

例 2: この例では、単ルート I/O 仮想化 (SR-IOV) ネットワークサポートパラメータ - SrioNetSupport の値に「simple」を指定して、指定されたインスタンスの拡張ネットワーキングを有効にします。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SrioNetSupport "simple"
```

例 3: この例では、指定されたインスタンスのセキュリティグループを変更します。インスタンスは VPC にある必要があります。名前ではなく、各セキュリティグループの ID を指定する必要があります。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -Group @( "sg-12345678",  
"sg-45678901" )
```

例 4: この例では、指定されたインスタンスの EBS I/O 最適化を有効にします。この機能は、すべてのインスタンスタイプで利用できるわけではありません。EBS 最適化インスタンスを使用する場合は、追加料金が適用されます。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -EbsOptimized $true
```

例 5: この例では、指定されたインスタンスの送信元/送信先チェックを有効にします。NAT インスタンスが NAT を実行するには、値が「false」である必要があります。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SourceDestCheck $true
```

例 6: この例では、指定されたインスタンスの終了を無効にします。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -DisableApiTermination $true
```

例 7: この例では、インスタンスからシャットダウンを開始したときに終了するように、指定されたインスタンスを変更します。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -  
InstanceInitiatedShutdownBehavior terminate
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[ModifyInstanceAttribute](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **ModifyInstanceCreditSpecification** を使用する

次のサンプルコードは、ModifyInstanceCreditSpecification を使用する方法を説明しています。

CLI

AWS CLI

インスタンスの CPU 使用率に対するクレジットオプションを変更するには

この例では、指定されたリージョン内にある指定されたインスタンスの CPU 使用率のクレジットオプションを「無制限」に変更します。有効なクレジットオプションは、「標準」と「無制限」です。

コマンド:

```
aws ec2 modify-instance-credit-specification --instance-credit-  
specification "InstanceId=i-1234567890abcdef0,CpuCredits=unlimited"
```

出力:

```
{
  "SuccessfulInstanceCreditSpecifications": [
    {
      "InstanceId": "i-1234567890abcdef0"
    }
  ],
  "UnsuccessfulInstanceCreditSpecifications": []
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ModifyInstanceCreditSpecification](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: これにより、インスタンス i-01234567890abcdef の T2 無制限クレジットが有効になります。

```
$Credit = New-Object -TypeName
    Amazon.EC2.Model.InstanceCreditSpecificationRequest
$Credit.InstanceId = "i-01234567890abcdef"
$Credit.CpuCredits = "unlimited"
Edit-EC2InstanceCreditSpecification -InstanceCreditSpecification $Credit
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ModifyInstanceCreditSpecification](#)」を参照してください。

Tools for PowerShell V5

例 1: これにより、インスタンス i-01234567890abcdef の T2 無制限クレジットが有効になります。

```
$Credit = New-Object -TypeName
    Amazon.EC2.Model.InstanceCreditSpecificationRequest
$Credit.InstanceId = "i-01234567890abcdef"
$Credit.CpuCredits = "unlimited"
Edit-EC2InstanceCreditSpecification -InstanceCreditSpecification $Credit
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の[ModifyInstanceCreditSpecification](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で `ModifyNetworkInterfaceAttribute` を使用する

次のサンプルコードは、`ModifyNetworkInterfaceAttribute` を使用方法を説明しています。

CLI

AWS CLI

ネットワークインターフェイスのアタッチメント属性を変更するには

このコマンド例では、指定されたネットワークインターフェイスの `attachment` 属性を変更します。

コマンド:

```
aws ec2 modify-network-interface-attribute --network-interface-id eni-686ea200 --attachment AttachmentId=eni-attach-43348162,DeleteOnTermination=false
```

ネットワークインターフェイスの説明属性を変更するには

このコマンド例では、指定されたネットワークインターフェイスの `description` 属性を変更します。

コマンド:

```
aws ec2 modify-network-interface-attribute --network-interface-id eni-686ea200 --description "My description"
```

ネットワークインターフェイスの `groupSet` 属性を変更するには

このコマンド例では、指定されたネットワークインターフェイスの `groupSet` 属性を変更します。

コマンド:

```
aws ec2 modify-network-interface-attribute --network-interface-id eni-686ea200 --groups sg-903004f8 sg-1a2b3c4d
```

ネットワークインターフェイスの `sourceDestCheck` 属性を変更するには

このコマンド例では、指定されたネットワークインターフェイスの `sourceDestCheck` 属性を変更します。

コマンド:

```
aws ec2 modify-network-interface-attribute --network-interface-id eni-686ea200 --no-source-dest-check
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ModifyNetworkInterfaceAttribute](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定された添付ファイルが終了時に削除されるように、指定されたネットワークインターフェイスを変更します。

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Attachment_AttachmentId eni-attach-1a2b3c4d -Attachment_DeleteOnTermination $true
```

例 2: この例では、指定されたネットワークインターフェイスの説明を変更します。

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Description "my description"
```

例 3: この例では、指定されたネットワークインターフェイスのセキュリティグループを変更します。

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Groups sg-1a2b3c4d
```

例 4: この例では、指定されたネットワークインターフェイスの送信元/送信先チェックを無効にします。

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -SourceDestCheck $false
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ModifyNetworkInterfaceAttribute](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定された添付ファイルが終了時に削除されるように、指定されたネットワークインターフェイスを変更します。

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -  
Attachment_AttachmentId eni-attach-1a2b3c4d -Attachment_DeleteOnTermination $true
```

例 2: この例では、指定されたネットワークインターフェイスの説明を変更します。

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Description  
"my description"
```

例 3: この例では、指定されたネットワークインターフェイスのセキュリティグループを変更します。

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Groups  
sg-1a2b3c4d
```

例 4: この例では、指定されたネットワークインターフェイスの送信元/送信先チェックを無効にします。

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -  
SourceDestCheck $false
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の[ModifyNetworkInterfaceAttribute](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **ModifyReservedInstances** を使用する

次のサンプルコードは、**ModifyReservedInstances** を使用方法を説明しています。

CLI

AWS CLI

リザーブドインスタンスを変更するには

このコマンド例では、リザーブドインスタンスを同じリージョンにある別のアベイラビリティゾーンに移動します。

コマンド:

```
aws ec2 modify-reserved-instances --reserved-instances-ids b847fa93-e282-4f55-b59a-1342f5bd7c02 --target-configurations AvailabilityZone=us-west-1c,Platform=EC2-Classic,InstanceCount=10
```

出力:

```
{
  "ReservedInstancesModificationId": "rimod-d3ed4335-b1d3-4de6-ab31-0f13aaf46687"
}
```

リザーブドインスタンスのネットワークプラットフォームを変更するには

このコマンド例では、EC2-Classic リザーブドインスタンスを EC2-VPC に変換します。

コマンド:

```
aws ec2 modify-reserved-instances --reserved-instances-ids f127bd27-edb7-44c9-a0eb-0d7e09259af0 --target-configurations AvailabilityZone=us-west-1c,Platform=EC2-VPC,InstanceCount=5
```

出力:

```
{
  "ReservedInstancesModificationId": "rimod-82fa9020-668f-4fb6-945d-61537009d291"
}
```

詳細については、「Amazon EC2 ユーザーガイド」の「リザーブドインスタンスの変更」を参照してください。

リザーブドインスタンスのインスタンスサイズを変更するには

このコマンド例では、us-west-1c に 10 個の m1.small Linux/UNIX インスタンスを持つリザーブドインスタンスを変更して、8 個の m1.small インスタンスを 2 個の m1.large インスタンスにし、残りの 2 個の m1.small を同じアベイラビリティゾーンの 1 個の m1.medium インスタンスにします。コマンド:

```
aws ec2 modify-reserved-instances --reserved-instances-ids 1ba8e2e3-3556-4264-949e-63ee671405a9 --target-configurations AvailabilityZone=us-west-1c,Platform=EC2-Classic,InstanceCount=2,InstanceType=m1.large AvailabilityZone=us-west-1c,Platform=EC2-Classic,InstanceCount=1,InstanceType=m1.medium
```

出力:

```
{
  "ReservedInstancesModificationId": "rimod-acc5f240-080d-4717-b3e3-1c6b11fa00b6"
}
```

詳細については、「Amazon EC2 ユーザーガイド」の「Modifying the Instance Size of Your Reservations」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ModifyReservedInstances](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたリザーブドインスタンスのアベイラビリティゾーン、インスタンス数、プラットフォームを変更します。

```
$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
```


- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ModifyReservedInstances](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたリザーブドインスタンスのアベイラビリティゾーン、インスタンス数、プラットフォームを変更します。

```
$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
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[ModifyReservedInstances](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **ModifySnapshotAttribute** を使用する

次のサンプルコードは、**ModifySnapshotAttribute** を使用する方法を説明しています。

CLI

AWS CLI

例 1: スナップショット属性を変更するには

次の `modify-snapshot-attribute` の例では、指定したスナップショットの `createVolumePermission` 属性を更新し、指定したユーザーのボリュームアクセス許可を削除します。

```
aws ec2 modify-snapshot-attribute \
--snapshot-id snap-1234567890abcdef0 \
```

```
--attribute createVolumePermission \  
--operation-type remove \  
--user-ids 123456789012
```

例 2: スナップショットをパブリックにするには

次の `modify-snapshot-attribute` の例では、指定したスナップショットを公開します。

```
aws ec2 modify-snapshot-attribute \  
--snapshot-id snap-1234567890abcdef0 \  
--attribute createVolumePermission \  
--operation-type add \  
--group-names all
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ModifySnapshotAttribute](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、`CreateVolumePermission` 属性を設定して、指定されたスナップショットを公開します。

```
Edit-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute  
CreateVolumePermission -OperationType Add -GroupName all
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ModifySnapshotAttribute](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、`CreateVolumePermission` 属性を設定して、指定されたスナップショットを公開します。

```
Edit-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute  
CreateVolumePermission -OperationType Add -GroupName all
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[ModifySnapshotAttribute](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **ModifySpotFleetRequest** を使用する

次のサンプルコードは、ModifySpotFleetRequest を使用方法を説明しています。

CLI

AWS CLI

スポットフリートリクエストを変更するには

このコマンド例では、指定されたスポットフリートリクエストのターゲット容量を更新します。

コマンド:

```
aws ec2 modify-spot-fleet-request --target-capacity 20 --spot-fleet-request-id sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
```

出力:

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

このコマンド例では、結果的にいずれのスポットインスタンスも終了せずに、指定したスポットフリートのターゲット容量を減らします。

コマンド:

```
aws ec2 modify-spot-fleet-request --target-capacity 10 --excess-capacity-termination-policy NoTermination --spot-fleet-request-ids sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
```

出力:

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

```
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ModifySpotFleetRequest](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: このコマンド例では、指定されたスポットフリートリクエストのターゲット容量を更新します。

```
Edit-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE -TargetCapacity 10
```

出力:

```
True
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ModifySpotFleetRequest](#)」を参照してください。

Tools for PowerShell V5

例 1: このコマンド例では、指定されたスポットフリートリクエストのターゲット容量を更新します。

```
Edit-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE -TargetCapacity 10
```

出力:

```
True
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[ModifySpotFleetRequest](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください [AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で `ModifySubnetAttribute` を使用する

次のサンプルコードは、`ModifySubnetAttribute` を使用方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [Amazon VPC の使用を開始する](#)

CLI

AWS CLI

サブネットのパブリック IPv4 アドレス指定の動作を変更するには

この例では、`subnet-1a2b3c4d` を変更して、このサブネットで起動されたすべてのインスタンスにパブリック IPv4 アドレスが割り当てられるように指定します。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 modify-subnet-attribute --subnet-id subnet-1a2b3c4d --map-public-ip-on-launch
```

サブネットの IPv6 アドレス指定の動作を変更するには

この例では、`subnet-1a2b3c4d` を変更して、このサブネットで起動されたすべてのインスタンスに、そのサブネットの範囲から IPv6 アドレスが割り当てられるように指定します。

コマンド:

```
aws ec2 modify-subnet-attribute --subnet-id subnet-1a2b3c4d --assign-ipv6-address-on-creation
```

詳細については、「AWS Virtual Private Cloud ユーザーガイド」の「VPC とサブネットの IP アドレス指定」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ModifySubnetAttribute](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたサブネットのパブリック IP アドレス指定を有効にします。

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $true
```

例 2: この例では、指定されたサブネットのパブリック IP アドレス指定を無効にします。

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $false
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ModifySubnetAttribute](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたサブネットのパブリック IP アドレス指定を有効にします。

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $true
```

例 2: この例では、指定されたサブネットのパブリック IP アドレス指定を無効にします。

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $false
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[ModifySubnetAttribute](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください [AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **ModifyVolumeAttribute** を使用する

次のサンプルコードは、ModifyVolumeAttribute を使用方法を説明しています。

CLI

AWS CLI

ボリューム属性を変更するには

この例では、ID `vol-1234567890abcdef0` を持つボリュームの `autoEnableIo` 属性を `true` に設定します。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 modify-volume-attribute --volume-id vol-1234567890abcdef0 --auto-enable-io
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ModifyVolumeAttribute](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたボリュームの指定された属性を変更します。ボリュームの I/O オペレーションは、データの不整合の可能性が原因で中断された後に自動的に再開されます。

```
Edit-EC2VolumeAttribute -VolumeId vol-12345678 -AutoEnableIO $true
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ModifyVolumeAttribute](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたボリュームの指定された属性を変更します。ボリュームの I/O オペレーションは、データの不整合の可能性が原因で中断された後に自動的に再開されます。

```
Edit-EC2VolumeAttribute -VolumeId vol-12345678 -AutoEnableIO $true
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[ModifyVolumeAttribute](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください [AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で `ModifyVpcAttribute` を使用する

次のサンプルコードは、`ModifyVpcAttribute` を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [Amazon VPC の使用を開始する](#)

CLI

AWS CLI

enableDnsSupport 属性を変更するには

この例では、enableDnsSupport 属性を変更します。この属性は、VPC に対して DNS 解決が有効になっているかどうかを示します。この属性が true の場合、Amazon DNS サーバーはインスタンスの DNS ホスト名を対応する IP アドレスに解決します。それ以外の場合は解決しません。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 modify-vpc-attribute --vpc-id vpc-a01106c2 --enable-dns-support "{\"Value\":false}"
```

enableDnsHostnames 属性を変更するには

この例では、enableDnsHostnames 属性を変更します。この属性は、VPC 内に起動されるインスタンスが DNS ホスト名を取得するかどうかを示します。この属性が true の場合、VPC 内のインスタンスは DNS ホスト名を取得します。それ以外の場合は取得しません。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 modify-vpc-attribute --vpc-id vpc-a01106c2 --enable-dns-hostnames "{\"Value\":false}"
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ModifyVpcAttribute](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定された VPC の DNS ホスト名のサポートを有効にします。


```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $true
```

例 2: この例では、指定された VPC の DNS ホスト名のサポートを無効にします。

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $false
```

例 3: この例では、指定された VPC の DNS 解決のサポートを有効にします。

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $true
```

例 4: この例では、指定された VPC の DNS 解決のサポートを無効にします。

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $false
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ModifyVpcAttribute](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定された VPC の DNS ホスト名のサポートを有効にします。

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $true
```

例 2: この例では、指定された VPC の DNS ホスト名のサポートを無効にします。

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $false
```

例 3: この例では、指定された VPC の DNS 解決のサポートを有効にします。

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $true
```

例 4: この例では、指定された VPC の DNS 解決のサポートを無効にします。

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $false
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[ModifyVpcAttribute](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **MonitorInstances**で を使用する

次のサンプルコードは、MonitorInstances を使用する方法を説明しています。

C++

SDK for C++

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
//! Enable detailed monitoring for an Amazon Elastic Compute Cloud (Amazon EC2)
instance.
/*!
    \param instanceId: An EC2 instance ID.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::EC2::enableMonitoring(const Aws::String &instanceId,
                                   const Aws::Client::ClientConfiguration
                                   &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::MonitorInstancesRequest request;
    request.AddInstanceIds(instanceId);
    request.SetDryRun(true);

    Aws::EC2::Model::MonitorInstancesOutcome dryRunOutcome =
    ec2Client.MonitorInstances(request);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
            << "Failed dry run to enable monitoring on instance. A dry run
            should trigger an error."
            <<
            std::endl;
        return false;
    }
```

```

    } else if (dryRunOutcome.GetError().GetErrorType()
               != Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cerr << "Failed dry run to enable monitoring on instance " <<
            instanceId << ": " << dryRunOutcome.GetError().GetMessage() <<
            std::endl;
        return false;
    }

    request.SetDryRun(false);
    Aws::EC2::Model::MonitorInstancesOutcome monitorInstancesOutcome =
ec2Client.MonitorInstances(request);
    if (!monitorInstancesOutcome.IsSuccess()) {
        std::cerr << "Failed to enable monitoring on instance " <<
            instanceId << ": " <<
            monitorInstancesOutcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully enabled monitoring on instance " <<
            instanceId << std::endl;
    }

    return monitorInstancesOutcome.IsSuccess();
}

```

- APIの詳細については、AWS SDK for C++ API リファレンスの「[MonitorInstances](#)」を参照してください。

CLI

AWS CLI

インスタンスの詳細モニタリングを有効にするには

このコマンド例は、指定されたインスタンスの詳細モニタリングを有効にします。

コマンド:

```
aws ec2 monitor-instances --instance-ids i-1234567890abcdef0
```

出力:

```
{
```

```

    "InstanceMonitorings": [
      {
        "InstanceId": "i-1234567890abcdef0",
        "Monitoring": {
          "State": "pending"
        }
      }
    ]
  }
}

```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[MonitorInstances](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

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

/**
 * Turn on detailed monitoring for the selected instance.
 * By default, metrics are sent to Amazon CloudWatch every 5 minutes.
 * For a cost you can enable detailed monitoring which sends metrics every
 * minute.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new MonitorInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { InstanceMonitorings } = await client.send(command);
    const instancesBeingMonitored = InstanceMonitorings.map(

```

```

    (im) =>
      ` • Detailed monitoring state for ${im.InstanceId} is
    ${im.Monitoring.State}.`,
    );
    console.log("Monitoring status:");
    console.log(instancesBeingMonitored.join("\n"));
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};

```

- API の詳細については、AWS SDK for JavaScript API リファレンスの「[MonitorInstances](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたインスタンスの詳細モニタリングを有効にします。

```
Start-EC2InstanceMonitoring -InstanceId i-12345678
```

出力:

InstanceId	Monitoring
-----	-----
i-12345678	Amazon.EC2.Model.Monitoring

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[MonitorInstances](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたインスタンスの詳細モニタリングを有効にします。

```
Start-EC2InstanceMonitoring -InstanceId i-12345678
```

出力:

InstanceId	Monitoring
-----	-----
i-12345678	Amazon.EC2.Model.Monitoring

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[MonitorInstances](#)」を参照してください。

SAP ABAP

SDK for SAP ABAP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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.

```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「[MonitorInstances](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **MoveAddressToVpc** を使用する

次のサンプルコードは、MoveAddressToVpc を使用する方法を説明しています。

CLI

AWS CLI

アドレスを EC2-VPC に移動するには

この例では、Elastic IP アドレス 54.123.4.56 を EC2-VPC プラットフォームに移動します。

コマンド:

```
aws ec2 move-address-to-vpc --public-ip 54.123.4.56
```

出力:

```
{
  "Status": "MoveInProgress"
}
```

```
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[MoveAddressToVpc](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、パブリック IP アドレスが 12.345.67.89 の EC2 インスタンスを、米国東部 (バージニア北部) リージョンの EC2-VPC プラットフォームに移動します。

```
Move-EC2AddressToVpc -PublicIp 12.345.67.89 -Region us-east-1
```

例 2: この例では、Get-EC2Instance コマンドの結果を Move-EC2AddressToVpc コマンドレットにパイプします。Get-EC2Instance コマンドは、インスタンス ID で指定されたインスタンスを取得し、インスタンスのパブリック IP アドレスプロパティを返します。

```
(Get-EC2Instance -Instance i-12345678).Instances.PublicIpAddress | Move-EC2AddressToVpc
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[MoveAddressToVpc](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、パブリック IP アドレスが 12.345.67.89 の EC2 インスタンスを、米国東部 (バージニア北部) リージョンの EC2-VPC プラットフォームに移動します。

```
Move-EC2AddressToVpc -PublicIp 12.345.67.89 -Region us-east-1
```

例 2: この例では、Get-EC2Instance コマンドの結果を Move-EC2AddressToVpc コマンドレットにパイプします。Get-EC2Instance コマンドは、インスタンス ID で指定されたインスタンスを取得し、インスタンスのパブリック IP アドレスプロパティを返します。

```
(Get-EC2Instance -Instance i-12345678).Instances.PublicIpAddress | Move-EC2AddressToVpc
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[MoveAddressToVpc](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **PurchaseHostReservation** を使用する

次のサンプルコードは、PurchaseHostReservation を使用する方法を説明しています。

CLI

AWS CLI

専有ホストの予約を購入するには

この例では、アカウント内の指定された専有ホストに対して指定された専有ホスト予約サービスを購入します。

コマンド:

```
aws ec2 purchase-host-reservation --offering-id hro-03f707bf363b6b324 --host-id-set h-013abcd2a00cbd123
```

出力:

```
{
  "TotalHourlyPrice": "1.499",
  "Purchase": [
    {
      "HourlyPrice": "1.499",
      "InstanceFamily": "m4",
      "PaymentOption": "NoUpfront",
      "HostIdSet": [
        "h-013abcd2a00cbd123"
      ],
      "HostReservationId": "hr-0d418a3a4ffc669ae",
      "UpfrontPrice": "0.000",
      "Duration": 31536000
    }
  ],
  "TotalUpfrontPrice": "0.000"
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[PurchaseHostReservation](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、専有ホスト h-01e23f4cd567890f1 の構成と一致する設定で、hro-0c1f23456789d0ab の予約サービスを購入します。

```
New-EC2HostReservation -OfferingId hro-0c1f23456789d0ab HostIdSet  
h-01e23f4cd567890f1
```

出力:

```
ClientToken      :  
CurrencyCode     :  
Purchase         : {hr-0123f4b5d67bedc89}  
TotalHourlyPrice : 1.307  
TotalUpfrontPrice : 0.000
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[PurchaseHostReservation](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、専有ホスト h-01e23f4cd567890f1 の構成と一致する設定で、hro-0c1f23456789d0ab の予約サービスを購入します。

```
New-EC2HostReservation -OfferingId hro-0c1f23456789d0ab HostIdSet  
h-01e23f4cd567890f1
```

出力:

```
ClientToken      :  
CurrencyCode     :  
Purchase         : {hr-0123f4b5d67bedc89}  
TotalHourlyPrice : 1.307  
TotalUpfrontPrice : 0.000
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[PurchaseHostReservation](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **PurchaseScheduledInstances** を使用する

次のサンプルコードは、PurchaseScheduledInstances を使用する方法を説明しています。

CLI

AWS CLI

スケジュールされたインスタンスを購入するには

この例では、スケジュールされたインスタンスを購入します。

コマンド:

```
aws ec2 purchase-scheduled-instances --purchase-requests file://purchase-request.json
```

Purchase-request.json:

```
[
  {
    "PurchaseToken": "eyJ2IjoiMSIsInMiOiEsImMiOi...",
    "InstanceCount": 1
  }
]
```

出力:

```
{
  "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"
  }
]
}

```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[PurchaseScheduledInstances](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、スケジュールされたインスタンスを購入します。

```

$request = New-Object Amazon.EC2.Model.PurchaseRequest
$request.InstanceCount = 1
$request.PurchaseToken = "eyJ2IjoiMSIsInMiOiEsImMiOi..."
New-EC2ScheduledInstancePurchase -PurchaseRequest $request

```

出力:

```

AvailabilityZone      : us-west-2b
CreateDate            : 1/25/2016 1:43:38 PM

```

```

HourlyPrice           : 0.095
InstanceCount         : 1
InstanceType          : c4.large
NetworkPlatform       : EC2-VPC
NextSlotStartTime     : 1/31/2016 1:00:00 AM
Platform              : Linux/UNIX
PreviousSlotEndTime   :
Recurrence             : Amazon.EC2.Model.ScheduledInstanceRecurrence
ScheduledInstanceId   : sci-1234-1234-1234-1234-123456789012
SlotDurationInHours   : 32
TermEndDate           : 1/31/2017 1:00:00 AM
TermStartDate         : 1/31/2016 1:00:00 AM
TotalScheduledInstanceHours : 1696

```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[PurchaseScheduledInstances](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、スケジュールされたインスタンスを購入します。

```

$request = New-Object Amazon.EC2.Model.PurchaseRequest
$request.InstanceCount = 1
$request.PurchaseToken = "eyJ2IjoiMSIsInMiOjEsImMiOi..."
New-EC2ScheduledInstancePurchase -PurchaseRequest $request

```

出力:

```

AvailabilityZone       : us-west-2b
CreateDate             : 1/25/2016 1:43:38 PM
HourlyPrice            : 0.095
InstanceCount          : 1
InstanceType           : c4.large
NetworkPlatform        : EC2-VPC
NextSlotStartTime      : 1/31/2016 1:00:00 AM
Platform               : Linux/UNIX
PreviousSlotEndTime    :
Recurrence              : Amazon.EC2.Model.ScheduledInstanceRecurrence
ScheduledInstanceId     : sci-1234-1234-1234-1234-123456789012
SlotDurationInHours    : 32
TermEndDate            : 1/31/2017 1:00:00 AM
TermStartDate          : 1/31/2016 1:00:00 AM
TotalScheduledInstanceHours : 1696

```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[PurchaseScheduledInstances](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **RebootInstances**で を使用する

次のサンプルコードは、RebootInstances を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [レジリエントなサービスの構築と管理](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

ID でインスタンスを再起動します。

```
/// <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;
}

```

インスタンスのプロファイルを置き換えて再起動し、ウェブサーバーを再起動します。

```

    /// <summary>
    /// Replace the profile associated with a running instance. After the profile
    /// is replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
    /// ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate
    /// with the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association
    /// for the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
    credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()

```



```

        {
            Name = credsProfileName
        }
    });
    // Allow time before resetting.
    Thread.Sleep(25000);

    await _amazonEc2.RebootInstancesAsync(
        new RebootInstancesRequest(new List<string>() { instanceId }));
    Thread.Sleep(25000);
    var instanceReady = false;
    var retries = 5;
    while (retries-- > 0 && !instanceReady)
    {
        var instancesPaginator =
            _amazonSsm.Paginators.DescribeInstanceInformation(
                new DescribeInstanceInformationRequest());
        // Get the entire list using the paginator.
        await foreach (var instance in
instancesPaginator.InstanceInformationList)
        {
            instanceReady = instance.InstanceId == instanceId;
            if (instanceReady)
            {
                break;
            }
        }
    }
    Console.WriteLine("Waiting for instance to be running.");
    await WaitForInstanceState(instanceId, InstanceStateName.Running);
    Console.WriteLine("Instance ready.");
    Console.WriteLine($"Sending restart command to instance
{instanceId}");
    await _amazonSsm.SendCommandAsync(
        new SendCommandRequest()
        {
            InstanceIds = new List<string>() { instanceId },
            DocumentName = "AWS-RunShellScript",
            Parameters = new Dictionary<string, List<string>>()
            {
                {
                    "commands",
                    new List<string>() { "cd / && sudo python3 server.py
80" }

```

```

        }
    }
    });
    Console.WriteLine($"Restarted the web server on instance
{instanceId}");
}
catch (AmazonEC2Exception ec2Exception)
{
    if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
    {
        _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
    }

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

```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[RebootInstances](#)」を参照してください。

C++

SDK for C++

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

//! Reboot an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
\param instanceID: An EC2 instance ID.

```

```

\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::EC2::rebootInstance(const Aws::String &instanceId,
                                const Aws::Client::ClientConfiguration
                                &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::RebootInstancesRequest request;
    request.AddInstanceIds(instanceId);
    request.SetDryRun(true);

    Aws::EC2::Model::RebootInstancesOutcome dry_run_outcome =
ec2Client.RebootInstances(request);
    if (dry_run_outcome.IsSuccess()) {
        std::cerr
            << "Failed dry run to reboot on instance. A dry run should
trigger an error."
            <<
            std::endl;
        return false;
    } else if (dry_run_outcome.GetError().GetErrorType()
               != Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to reboot instance " << instanceId << ": "
            << dry_run_outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    request.SetDryRun(false);
    Aws::EC2::Model::RebootInstancesOutcome outcome =
ec2Client.RebootInstances(request);
    if (!outcome.IsSuccess()) {
        std::cout << "Failed to reboot instance " << instanceId << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully rebooted instance " << instanceId <<
            std::endl;
    }

    return outcome.IsSuccess();
}

```

- API の詳細については、AWS SDK for C++ API リファレンスの「[RebootInstances](#)」を参照してください。

CLI

AWS CLI

Amazon EC2 インスタンスを再起動するには

この例では、指定のインスタンスを再起動します。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 reboot-instances --instance-ids i-1234567890abcdef5
```

詳細については、「Amazon Elastic Compute Cloud ユーザーガイド」の「インスタンスの再起動方法」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[RebootInstances](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

/**
 * Requests a reboot of the specified instances. This operation is asynchronous;
 * it only queues a request to reboot the specified instances.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
```

```
const command = new RebootInstancesCommand({
  InstanceIds: instanceIds,
});

try {
  await client.send(command);
  console.log("Instance rebooted successfully.");
} catch (caught) {
  if (
    caught instanceof Error &&
    caught.name === "InvalidInstanceID.NotFound"
  ) {
    console.warn(
      `${caught.message}. Please provide the InstanceId of a valid instance to
      reboot.`
    );
  } else {
    throw caught;
  }
}
```

- API の詳細については、AWS SDK for JavaScript API リファレンスの「[RebootInstances](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたインスタンスを再起動します。

```
Restart-EC2Instance -InstanceId i-12345678
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[RebootInstances](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたインスタンスを再起動します。

```
Restart-EC2Instance -InstanceId i-12345678
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[RebootInstances](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
```

```

self.inst_type = inst_type
self.ami_param = ami_param
self.autoscaling_client = autoscaling_client
self.ec2_client = ec2_client
self.ssm_client = ssm_client
self.iam_client = iam_client
sts_client = boto3.client("sts")
self.account_id = sts_client.get_caller_identity()["Account"]

self.key_pair_name = f"{resource_prefix}-key-pair"
self.launch_template_name = f"{resource_prefix}-template-"
self.group_name = f"{resource_prefix}-group"

# Happy path
self.instance_policy_name = f"{resource_prefix}-pol"
self.instance_role_name = f"{resource_prefix}-role"
self.instance_profile_name = f"{resource_prefix}-prof"

# Failure mode
self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
self.bad_creds_role_name = f"{resource_prefix}-bc-role"
self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

def replace_instance_profile(
    self,
    instance_id: str,
    new_instance_profile_name: str,
    profile_association_id: str,
) -> None:
    """
    Replaces the profile associated with a running instance. After the
    profile is
    replaced, the instance is rebooted to ensure that it uses the new
    profile. When
    the instance is ready, Systems Manager is used to restart the Python web
    server.

    :param instance_id: The ID of the instance to restart.
    :param new_instance_profile_name: The name of the new profile to
    associate with
    the specified instance.
    :param profile_association_id: The ID of the existing profile association
    for the

```

```

instance.

"""
try:
    self.ec2_client.replace_iam_instance_profile_association(
        IamInstanceProfile={"Name": new_instance_profile_name},
        AssociationId=profile_association_id,
    )
    log.info(
        "Replaced instance profile for association %s with profile %s.",
        profile_association_id,
        new_instance_profile_name,
    )
    time.sleep(5)

    self.ec2_client.reboot_instances(InstanceIds=[instance_id])
    log.info("Rebooting instance %s.", instance_id)
    waiter = self.ec2_client.get_waiter("instance_running")
    log.info("Waiting for instance %s to be running.", instance_id)
    waiter.wait(InstanceIds=[instance_id])
    log.info("Instance %s is now running.", instance_id)

    self.ssm_client.send_command(
        InstanceIds=[instance_id],
        DocumentName="AWS-RunShellScript",
        Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
    )
    log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
except ClientError as err:
    log.error("Failed to replace instance profile.")
    error_code = err.response["Error"]["Code"]
    if error_code == "InvalidAssociationID.NotFound":
        log.error(
            f"Association ID '{profile_association_id}' does not exist."
            "Please check the association ID and try again."
        )
    if error_code == "InvalidInstanceId":
        log.error(
            f"The specified instance ID '{instance_id}' does not exist or
is not available for SSM. "
            f"Please verify the instance ID and try again."
        )
    log.error(f"Full error:\n\t{err}")

```


- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[RebootInstances](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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(())
}
```

Writers API を使用して、インスタンスが停止状態および準備完了状態になるのを待ちます。Writers API を使用するには、rust ファイルで `use aws\_sdk\_ec2::client::Writers` を使用する必要があります。

```
/// 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(())
}
```

```
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[RebootInstances](#)」を参照してください。

SAP ABAP

SDK for SAP ABAP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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.
```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「[RebootInstances](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **RegisterImage** を使用する

次のサンプルコードは、RegisterImage を使用する方法を説明しています。

CLI

AWS CLI

例 1: マニフェストファイルを使用して AMI を登録するには

次の register-image の例では、Amazon S3 で指定されたマニフェストファイルを使用して AMI を登録します。

```
aws ec2 register-image \
  --name my-image \
  --image-location amzn-s3-demo-bucket/myimage/image.manifest.xml
```

出力:

```
{
  "ImageId": "ami-1234567890EXAMPLE"
}
```

詳細については「アマゾン EC2 ユーザーガイド」の「[アマゾン マシンイメージ \(AMI\)](#)」を参照してください。

例 2: ルートデバイスのスナップショットを使用して AMI を登録するには

次の `register-image` の例では、EBS ルートボリュームの指定されたスナップショットをデバイス `/dev/xvda` として使用して、AMI を登録します。ブロックデバイスマッピングには、デバイス `/dev/xvdf` として空の 100 GiB EBS ボリュームも含まれます。

```
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}
```

出力:

```
{  
  "ImageId": "ami-1a2b3c4d5eEXAMPLE"  
}
```

詳細については「アマゾン EC2 ユーザーガイド」の「[アマゾン マシンイメージ \(AMI\)](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[RegisterImage](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、Amazon S3 の指定されたマニフェストファイルを使用して AMI を登録します。

```
Register-EC2Image -ImageLocation amzn-s3-demo-bucket/my-web-server-ami/  
image.manifest.xml -Name my-web-server-ami
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[RegisterImage](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、Amazon S3 の指定されたマニフェストファイルを使用して AMI を登録します。

```
Register-EC2Image -ImageLocation amzn-s3-demo-bucket/my-web-server-ami/  
image.manifest.xml -Name my-web-server-ami
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[RegisterImage](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **RejectVpcPeeringConnection** を使用する

次のサンプルコードは、RejectVpcPeeringConnection を使用する方法を説明しています。

CLI

AWS CLI

VPC ピアリング接続を拒否するには

この例では、指定された VPC ピアリング接続リクエストを拒否します。

コマンド:

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

出力:

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

- API の詳細については、「AWS CLI コマンドリファレンス」の「[RejectVpcPeeringConnection](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: 上記の例では、VpcPeering リクエスト ID pcx-01a2b3ce45fe67eb8 のリクエストを拒否します。

```
Deny-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-01a2b3ce45fe67eb8
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[RejectVpcPeeringConnection](#)」を参照してください。

Tools for PowerShell V5

例 1: 上記の例では、VpcPeering リクエスト ID pcx-01a2b3ce45fe67eb8 のリクエストを拒否します。

```
Deny-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-01a2b3ce45fe67eb8
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[RejectVpcPeeringConnection](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **ReleaseAddress**で を使用する

次のサンプルコードは、ReleaseAddress を使用方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)
- [プライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成する](#)
- [Amazon VPC の使用を開始する](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

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

        return false;
    }
    catch (Exception ex)
    {
        _logger.LogError(
```



```

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

```

- API の詳細については、AWS SDK for .NET API リファレンスの「[ReleaseAddress](#)」を参照してください。

Bash

AWS CLI Bash スクリプトを使用する

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

#####
# 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
}
```

この例で使用されているユーティリティ関数。

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

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ReleaseAddress](#)」を参照してください。

C++

SDK for C++

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
#!/ Release an Elastic IP address.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::releaseAddress(const Aws::String &allocationID,
                                const Aws::Client::ClientConfiguration
                                &clientConfiguration) {
    Aws::EC2::EC2Client ec2(clientConfiguration);

    Aws::EC2::Model::ReleaseAddressRequest request;
    request.SetAllocationId(allocationID);

    Aws::EC2::Model::ReleaseAddressOutcome outcome = ec2.ReleaseAddress(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to release Elastic IP address " <<
                    allocationID << ":" << outcome.GetError().GetMessage() <<
                    std::endl;
    } else {
        std::cout << "Successfully released Elastic IP address " <<
                    allocationID << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API の詳細については、AWS SDK for C++ API リファレンスの「[ReleaseAddress](#)」を参照してください。

CLI

AWS CLI

EC2-Classic 用 Elastic IP アドレスをリリースするには

この例では、EC2-Classic のインスタンスに使用する Elastic IP アドレスをリリースしています。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 release-address --public-ip 198.51.100.0
```

EC2-VPC 用 Elastic IP アドレスをリリースするには

この例では、VPC のインスタンスに使用する Elastic IP アドレスをリリースしています。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 release-address --allocation-id eipalloc-64d5890a
```

- API の詳細については、「AWS CLI Command Reference」の「[ReleaseAddress](#)」を参照してください。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/**  
 * Releases an Elastic IP address asynchronously.
```

```

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

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

        return response;
    }

```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[ReleaseAddress](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

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

/**
 * Release an Elastic IP address.

```

```

* @param {{ allocationId: string }} options
*/
export const main = async ({ allocationId }) => {
  const client = new EC2Client({});
  const command = new ReleaseAddressCommand({
    // You can also use PublicIp, but that is for EC2 classic which is being
    retired.
    AllocationId: allocationId,
  });

  try {
    await client.send(command);
    console.log("Successfully released address.");
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidAllocationID.NotFound"
    ) {
      console.warn(`${caught.message}. Please provide a valid AllocationID.`);
    } else {
      throw caught;
    }
  }
};

```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[ReleaseAddress](#)」を参照してください。

Kotlin

SDK for Kotlin

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

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")  
}  
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[ReleaseAddress](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、VPC 内のインスタンスに指定された Elastic IP アドレスを解放します。

```
Remove-EC2Address -AllocationId eipalloc-12345678 -Force
```

例 2: この例では、EC2-Classic のインスタンスに指定された Elastic IP アドレスを解放します。

```
Remove-EC2Address -PublicIp 198.51.100.2 -Force
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ReleaseAddress](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、VPC 内のインスタンスに指定された Elastic IP アドレスを解放します。

```
Remove-EC2Address -AllocationId eipalloc-12345678 -Force
```

例 2: この例では、EC2-Classic のインスタンスに指定された Elastic IP アドレスを解放します。

```
Remove-EC2Address -PublicIp 198.51.100.2 -Force
```


- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[ReleaseAddress](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str]
            = None
        ) -> None:
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
            :param public_ip: The public IP address of the Elastic IP.
            :param instance_id: The ID of the associated EC2 instance, if any.
            """
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

        def __init__(self, ec2_client: Any) -> None:
            """
            Initializes the ElasticIpWrapper with an EC2 client.

            :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
            level
```

```

        access to AWS EC2 services.

        """
        self.ec2_client = ec2_client
        self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def release(self, allocation_id: str) -> None:
        """
        Releases an Elastic IP address. After the Elastic IP address is released,
it can no longer be used.

        :param allocation_id: The allocation ID of the Elastic IP to release.
        :raises ClientError: If the release fails, such as when the Elastic IP
address is not found.
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
        if elastic_ip is None:
            logger.info(f"No Elastic IP found with allocation ID
{allocation_id}.")
            return

        try:
            self.ec2_client.release_address(AllocationId=allocation_id)
            self.elastic_ips.remove(elastic_ip) # Remove the Elastic IP from the
list
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidAddress.NotFound":
                logger.error(
                    f"Failed to release Elastic IP address {allocation_id} "
                    "because it could not be found. Verify the Elastic IP address
                    "

```

```

        "and ensure it is allocated to your account in the correct
region "
        "before attempting to release it."
    )
    raise

```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[ReleaseAddress](#)」を参照してください。

Ruby

SDK for Ruby

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

# 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
end

```

```
rescue StandardError => e
  puts("Error releasing Elastic IP address: #{e.message}")
  false
end
```

- API の詳細については、AWS SDK for Ruby API リファレンスの「[ReleaseAddress](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn deallocate_ip_address(&self, allocation_id: &str) -> Result<(),
EC2Error> {
    self.client
        .release_address()
        .allocation_id(allocation_id)
        .send()
        .await?;
    Ok(())
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[ReleaseAddress](#)」を参照してください。

SAP ABAP

SDK for SAP ABAP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
TRY.  
    lo_ec2->releaseaddress( iv_allocationid = iv_allocation_id ).  
    MESSAGE 'Elastic IP address released.' TYPE 'I'.  
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception->av_err_msg }|.  
    MESSAGE lv_error TYPE 'E'.  
ENDTRY.
```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「[ReleaseAddress](#)」を参照してください。

Swift

SDK for Swift

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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)")
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[ReleaseAddress](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **ReleaseHosts** を使用する

次のサンプルコードは、ReleaseHosts を使用する方法を説明しています。

CLI

AWS CLI

アカウントから専有ホストを解除するには

アカウントから専有ホストを解除するには。ホストを解除する前に、ホスト上にあるインスタンスを停止または終了する必要があります。

コマンド:

```
aws ec2 release-hosts --host-id=h-0029d6e3cacf1b3da
```

出力:

```
{
  "Successful": [
    "h-0029d6e3cacf1b3da"
  ]
}
```

```
    ],
    "Unsuccessful": []

}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ReleaseHosts](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたホスト ID h-0badafd1dcb2f3456 を解放します。

```
Remove-EC2Host -HostId h-0badafd1dcb2f3456
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Host (ReleaseHosts)" on target
"h-0badafd1dcb2f3456".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"): Y

Successful                Unsuccessful
-----
{h-0badafd1dcb2f3456} {}
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ReleaseHosts](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたホスト ID h-0badafd1dcb2f3456 を解放します。

```
Remove-EC2Host -HostId h-0badafd1dcb2f3456
```

出力:

```
Confirm
```

```
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Host (ReleaseHosts)" on target
"h-0badafd1dcb2f3456".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"): Y

Successful                Unsuccessful
-----
{h-0badafd1dcb2f3456} {}
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[ReleaseHosts](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください [AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI `ReplaceIamInstanceProfileAssociation`で を使用する

次のサンプルコードは、`ReplaceIamInstanceProfileAssociation` を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [レジリエントなサービスの構築と管理](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/// <summary>
/// Replace the profile associated with a running instance. After the profile
is replaced, the instance
```



```

    /// is rebooted to ensure that it uses the new profile. When the instance is
    ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate
    with the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association
    for the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
    credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
            // Allow time before resetting.
            Thread.Sleep(25000);

            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
                await foreach (var instance in
                    instancesPaginator.InstanceInformationList)
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {

```

```

        break;
    }
}
}
Console.WriteLine("Waiting for instance to be running.");
await WaitForInstanceState(instanceId, InstanceStateName.Running);
Console.WriteLine("Instance ready.");
Console.WriteLine($"Sending restart command to instance
{instanceId}");
await _amazonSsm.SendCommandAsync(
    new SendCommandRequest()
    {
        InstanceIds = new List<string>() { instanceId },
        DocumentName = "AWS-RunShellScript",
        Parameters = new Dictionary<string, List<string>>()
        {
            {
                "commands",
                new List<string>() { "cd / && sudo python3 server.py
80" }
            }
        }
    });
Console.WriteLine($"Restarted the web server on instance
{instanceId}");
}
catch (AmazonEC2Exception ec2Exception)
{
    if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
    {
        _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
    }

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

```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[ReplacelamInstanceProfileAssociation](#)」を参照してください。

CLI

AWS CLI

インスタンスの IAM インスタンスプロファイルを置き換えるには

この例では、関連付け iip-assoc-060bae234aac2e7fa によって表される IAM インスタンスプロファイルを、AdminRole という名前の IAM インスタンスプロファイルに置き換えています。

```
aws ec2 replace-iam-instance-profile-association \  
  --iam-instance-profile Name=AdminRole \  
  --association-id iip-assoc-060bae234aac2e7fa
```

出力:

```
{  
  "IamInstanceProfileAssociation": {  
    "InstanceId": "i-087711ddaf98f9489",  
    "State": "associating",  
    "AssociationId": "iip-assoc-0b215292fab192820",  
    "IamInstanceProfile": {  
      "Id": "AIPAJLNLDX3AMYZNWYYAY",  
      "Arn": "arn:aws:iam::123456789012:instance-profile/AdminRole"  
    }  
  }  
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ReplacelamInstanceProfileAssociation](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
  ec2Client.send(
    new ReplaceIamInstanceProfileAssociationCommand({
      AssociationId: state.instanceProfileAssociationId,
      IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
    }),
  ),
);
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[ReplacesIamInstanceProfileAssociation](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

この例では、実行中のインスタンスのインスタンスプロファイルを置き換え、インスタンスを再起動し、起動後にインスタンスにコマンドを送信します。

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

```

def __init__(
    self,
    resource_prefix: str,
    inst_type: str,
    ami_param: str,
    autoscaling_client: boto3.client,
    ec2_client: boto3.client,
    ssm_client: boto3.client,
    iam_client: boto3.client,
):
    """
    Initializes the AutoScaler class with the necessary parameters.

    :param resource_prefix: The prefix for naming AWS resources that are
    created by this class.
    :param inst_type: The type of EC2 instance to create, such as t3.micro.
    :param ami_param: The Systems Manager parameter used to look up the AMI
    that is created.
    :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
    :param ec2_client: A Boto3 EC2 client.
    :param ssm_client: A Boto3 Systems Manager client.
    :param iam_client: A Boto3 IAM client.
    """
    self.inst_type = inst_type
    self.ami_param = ami_param
    self.autoscaling_client = autoscaling_client
    self.ec2_client = ec2_client
    self.ssm_client = ssm_client
    self.iam_client = iam_client
    sts_client = boto3.client("sts")
    self.account_id = sts_client.get_caller_identity()["Account"]

    self.key_pair_name = f"{resource_prefix}-key-pair"
    self.launch_template_name = f"{resource_prefix}-template-"
    self.group_name = f"{resource_prefix}-group"

    # Happy path
    self.instance_policy_name = f"{resource_prefix}-pol"
    self.instance_role_name = f"{resource_prefix}-role"
    self.instance_profile_name = f"{resource_prefix}-prof"

    # Failure mode
    self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"

```

```

self.bad_creds_role_name = f"{resource_prefix}-bc-role"
self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

def replace_instance_profile(
    self,
    instance_id: str,
    new_instance_profile_name: str,
    profile_association_id: str,
) -> None:
    """
    Replaces the profile associated with a running instance. After the
    profile is
    replaced, the instance is rebooted to ensure that it uses the new
    profile. When
    the instance is ready, Systems Manager is used to restart the Python web
    server.

    :param instance_id: The ID of the instance to restart.
    :param new_instance_profile_name: The name of the new profile to
    associate with
                                the specified instance.
    :param profile_association_id: The ID of the existing profile association
    for the
                                instance.
    """
    try:
        self.ec2_client.replace_iam_instance_profile_association(
            IamInstanceProfile={"Name": new_instance_profile_name},
            AssociationId=profile_association_id,
        )
        log.info(
            "Replaced instance profile for association %s with profile %s.",
            profile_association_id,
            new_instance_profile_name,
        )
        time.sleep(5)

        self.ec2_client.reboot_instances(InstanceIds=[instance_id])
        log.info("Rebooting instance %s.", instance_id)
        waiter = self.ec2_client.get_waiter("instance_running")
        log.info("Waiting for instance %s to be running.", instance_id)
        waiter.wait(InstanceIds=[instance_id])
        log.info("Instance %s is now running.", instance_id)

```

```

        self.ssm_client.send_command(
            InstanceIds=[instance_id],
            DocumentName="AWS-RunShellScript",
            Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
        )
        log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
    except ClientError as err:
        log.error("Failed to replace instance profile.")
        error_code = err.response["Error"]["Code"]
        if error_code == "InvalidAssociationID.NotFound":
            log.error(
                f"Association ID '{profile_association_id}' does not exist."
                "Please check the association ID and try again."
            )
        if error_code == "InvalidInstanceId":
            log.error(
                f"The specified instance ID '{instance_id}' does not exist or
is not available for SSM. "
                f"Please verify the instance ID and try again."
            )
        log.error(f"Full error:\n\t{err}")

```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[ReplacelamInstanceProfileAssociation](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **ReplaceNetworkAclAssociation** を使用する

次のサンプルコードは、ReplaceNetworkAclAssociation を使用する方法を説明しています。

CLI

AWS CLI

ネットワーク ACL をサブネットと置き換えるには

この例では、指定されたネットワーク ACL を、指定されたネットワーク ACL 関連付けのサブネットに関連付けます。

コマンド:

```
aws ec2 replace-network-acl-association --association-id aclassoc-e5b95c8c --  
network-acl-id acl-5fb85d36
```

出力:

```
{  
  "NewAssociationId": "aclassoc-3999875b"  
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ReplaceNetworkAclAssociation](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたネットワーク ACL を、指定されたネットワーク ACL 関連付けを持つサブネットに関連付けます。

```
Set-EC2NetworkAclAssociation -NetworkAclId acl-12345678 -AssociationId  
aclassoc-1a2b3c4d
```

出力:

```
aclassoc-87654321
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ReplaceNetworkAclAssociation](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたネットワーク ACL を、指定されたネットワーク ACL 関連付けを持つサブネットに関連付けます。

```
Set-EC2NetworkAclAssociation -NetworkAclId acl-12345678 -AssociationId  
aclassoc-1a2b3c4d
```


出力:

```
aclassoc-87654321
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[ReplaceNetworkAclAssociation](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で `ReplaceNetworkAclEntry` を使用する

次のサンプルコードは、`ReplaceNetworkAclEntry` を使用する方法を説明しています。

CLI

AWS CLI

ネットワーク ACL エントリを置き換えるには

この例では、指定されたネットワーク ACL のエントリを置き換えます。新しいルール 100 は、UDP ポート 53 (DNS) の 203.0.113.12/24 から任意の関連付けられたサブネットへの入力トラフィックを許可します。

コマンド:

```
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
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ReplaceNetworkAclEntry](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたネットワーク ACL の指定されたエントリを置き換えます。新しいルールでは、指定されたアドレスから関連付けられた任意のサブネットへのインバウンドトラフィックが許可されます。

```
Set-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100  
-Protocol 17 -PortRange_From 53 -PortRange_To 53 -CidrBlock 203.0.113.12/24 -  
RuleAction allow
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ReplaceNetworkAclEntry](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたネットワーク ACL の指定されたエントリを置き換えます。新しいルールでは、指定されたアドレスから関連付けられた任意のサブネットへのインバウンドトラフィックが許可されます。

```
Set-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100  
-Protocol 17 -PortRange_From 53 -PortRange_To 53 -CidrBlock 203.0.113.12/24 -  
RuleAction allow
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[ReplaceNetworkAclEntry](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **ReplaceRoute** を使用する

次のサンプルコードは、ReplaceRoute を使用する方法を説明しています。

CLI

AWS CLI

ルートを置き換えるには

この例では、指定したルートテーブルの指定したルートを置き換えます。新しいルートは、指定された CIDR に一致し、トラフィックを指定された仮想プライベートゲートウェイに送信します。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 replace-route --route-table-id rtb-22574640 --destination-cidr-block 10.0.0.0/16 --gateway-id vgw-9a4cacf3
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ReplaceRoute](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定したルートテーブルの指定したルートを置き換えます。新しいルートは、指定されたトラフィックを指定された仮想プライベートゲートウェイに送信します。

```
Set-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 10.0.0.0/24 - GatewayId vgw-1a2b3c4d
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ReplaceRoute](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定したルートテーブルの指定したルートを置き換えます。新しいルートは、指定されたトラフィックを指定された仮想プライベートゲートウェイに送信します。

```
Set-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 10.0.0.0/24 - GatewayId vgw-1a2b3c4d
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[ReplaceRoute](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で `ReplaceRouteTableAssociation` を使用する

次のサンプルコードは、`ReplaceRouteTableAssociation` を使用方法を説明しています。

CLI

AWS CLI

サブネットに関連付けられたルートテーブルを置き換えるには

この例では、指定されたルートテーブルを、指定されたルートテーブルの関連付け用のサブネットに関連付けます。

コマンド:

```
aws ec2 replace-route-table-association --association-id rtbassoc-781d0d1a --  
route-table-id rtb-22574640
```

出力:

```
{  
  "NewAssociationId": "rtbassoc-3a1f0f58"  
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ReplaceRouteTableAssociation](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたルートテーブルを、指定されたルートテーブルの関連付けを持つサブネットに関連付けます。

```
Set-EC2RouteTableAssociation -RouteTableId rtb-1a2b3c4d -AssociationId  
rtbassoc-12345678
```

出力:

```
rtbassoc-87654321
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ReplaceRouteTableAssociation](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたルートテーブルを、指定されたルートテーブルの関連付けを持つサブネットに関連付けます。

```
Set-EC2RouteTableAssociation -RouteTableId rtb-1a2b3c4d -AssociationId  
rtbassoc-12345678
```

出力:

```
rtbassoc-87654321
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[ReplaceRouteTableAssociation](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **ReportInstanceStatus** を使用する

次のサンプルコードは、ReportInstanceStatus を使用する方法を説明しています。

CLI

AWS CLI

インスタンスのステータスフィードバックを報告するには

このコマンド例では、指定されたインスタンスのステータスフィードバックを報告します。

コマンド:

```
aws ec2 report-instance-status --instances i-1234567890abcdef0 --status impaired  
--reason-codes unresponsive
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ReportInstanceStatus](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたインスタンスのステータスフィードバックを報告します。

```
Send-EC2InstanceStatus -Instance i-12345678 -Status impaired -ReasonCode  
unresponsive
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ReportInstanceStatus](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたインスタンスのステータスフィードバックを報告します。

```
Send-EC2InstanceStatus -Instance i-12345678 -Status impaired -ReasonCode  
unresponsive
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[ReportInstanceStatus](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **RequestSpotFleet** を使用する

次のサンプルコードは、RequestSpotFleet を使用方法を説明しています。

CLI

AWS CLI

最低料金でサブネット内のスポットフリートをリクエストするには

このコマンド例では、サブネットによってのみ異なる 2 つの起動仕様を持つスポットフリートリクエストを作成します。スポットフリートは、指定された最低料金のサブネットでインスタンスを起動します。インスタンスがデフォルトの VPC で起動される場合は、デフォルトでパブリック IP アドレスが割り当てられます。インスタンスがデフォルト以外の VPC で起動される場合は、デフォルトでパブリック IP アドレスは割り当てられません。

スポットフリートリクエストで、同じアベイラビリティゾーンから異なるサブネットを指定することはできません。

コマンド:

```
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"
      }
    }
  ]
}
```

出力:

```
{
  "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE"
}
```

最低料金でアベイラビリティゾーンのスポットフリートをリクエストするには

このコマンド例では、アベイラビリティゾーンによってのみ異なる 2 つの起動仕様を持つスポットフリートリクエストを作成します。スポットフリートは、指定された最低料金のアベ

イラビリティーゾーンでインスタンスを起動します。アカウントが EC2-VPC のみをサポートしている場合、Amazon EC2 はアベイラビリティゾーンのデフォルトサブネットでスポットインスタンスを起動します。アカウントが EC2-Classic をサポートしている場合、Amazon EC2 はアベイラビリティゾーンの EC2-Classic でインスタンスを起動します。

コマンド:

```
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"
      }
    }
  ]
}
```

サブネットでスポットインスタンスを起動し、パブリック IP アドレスを割り当てるには

このコマンド例では、デフォルト以外の VPC で起動されたインスタンスにパブリックアドレスを割り当てます。ネットワークインターフェイスを指定する際、ネットワークインターフェイスを使用してサブネット ID とセキュリティグループ ID を含める必要があります。

コマンド:


```
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"
      }
    }
  ]
}
```

分散配分戦略を使用してスポットフリートをリクエストするには

このコマンド例では、分散配分戦略を使用して 30 個のインスタンスを起動するスポットフリートリクエストを作成します。起動仕様はインスタンスタイプによって異なります。スポットフリートは、各タイプのインスタンスが 10 個になるように、起動仕様全体にインスタンスを分散します。

コマンド:

```
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"
  }
]
```

詳細については、「Amazon Elastic Compute Cloud ユーザーガイド」の「Spot Fleet Requests」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[RequestSpotFleet](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定したインスタンスタイプが最低価格のアベイラビリティゾーンで、スポットフリートリクエストを作成します。アカウントが EC2-VPC のみをサポートしている場合、スポットフリートは、デフォルトのサブネットが存在する、最低料金のアベイラビリティゾーンでインスタンスを起動します。アカウントが EC2-Classic をサポートしている場合、スポットフリートは、最低料金のアベイラビリティゾーンの EC2-Classic でインスタンスを起動します。リクエストで指定したスポット料金を超えて課金されることはありません。

```
$sg = New-Object Amazon.EC2.Model.GroupIdentifier
```

```
$sg.GroupId = "sg-12345678"
$lc = New-Object Amazon.EC2.Model.SpotFleetLaunchSpecification
$lc.ImageId = "ami-12345678"
$lc.InstanceType = "m3.medium"
$lc.SecurityGroups.Add($sg)
Request-EC2SpotFleet -SpotFleetRequestConfig_SpotPrice 0.04 `
-SpotFleetRequestConfig_TargetCapacity 2 `
-SpotFleetRequestConfig_IamFleetRole arn:aws:iam::123456789012:role/my-spot-
fleet-role `
-SpotFleetRequestConfig_LaunchSpecification $lc
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[RequestSpotFleet](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定したインスタンスタイプが最低価格のアベイラビリティゾーンで、スポットフリートリクエストを作成します。アカウントが EC2-VPC のみをサポートしている場合、スポットフリートは、デフォルトのサブネットが存在する、最低料金のアベイラビリティゾーンでインスタンスを起動します。アカウントが EC2-Classic をサポートしている場合、スポットフリートは、最低料金のアベイラビリティゾーンの EC2-Classic でインスタンスを起動します。リクエストで指定したスポット料金を超えて課金されることはありません。

```
$sg = New-Object Amazon.EC2.Model.GroupIdentifier
$sg.GroupId = "sg-12345678"
$lc = New-Object Amazon.EC2.Model.SpotFleetLaunchSpecification
$lc.ImageId = "ami-12345678"
$lc.InstanceType = "m3.medium"
$lc.SecurityGroups.Add($sg)
Request-EC2SpotFleet -SpotFleetRequestConfig_SpotPrice 0.04 `
-SpotFleetRequestConfig_TargetCapacity 2 `
-SpotFleetRequestConfig_IamFleetRole arn:aws:iam::123456789012:role/my-spot-
fleet-role `
-SpotFleetRequestConfig_LaunchSpecification $lc
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[RequestSpotFleet](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で `RequestSpotInstances` を使用する

次のサンプルコードは、`RequestSpotInstances` を使用する方法を説明しています。

CLI

AWS CLI

スポットインスタンスをリクエストするには

このコマンド例では、指定されたアベイラビリティーゾーン内の 5 個のインスタンスに対して 1 回限りのスポットインスタンスリクエストを作成します。アカウントが EC2-VPC のみをサポートしている場合、Amazon EC2 は指定されたアベイラビリティーゾーンのデフォルトサブネットでインスタンスを起動します。アカウントが EC2-Classic をサポートしている場合、Amazon EC2 は指定されたアベイラビリティーゾーンの EC2-Classic でインスタンスを起動します。

コマンド:

```
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"  
  }  
}
```

出力:

```
{
  "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"
    },
    ...
  ]
}
```

このコマンド例では、指定されたサブネット内の 5 個のインスタンスに対して 1 回限りのスポットインスタンスリクエストを作成します。Amazon EC2 は指定されたサブネットでイン

スタンスを起動します。VPC がデフォルト以外の VPC である場合、インスタンスにはデフォルトでパブリック IP アドレスが割り当てられません。

コマンド:

```
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"  
  }  
}
```

出力:

```
{  
  "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"
},
...
]
}

```

この例では、デフォルト以外の VPC で起動するスポットインスタンスにパブリック IP アドレスを割り当てます。ネットワークインターフェイスを指定する際、ネットワークインターフェイスを使用してサブネット ID とセキュリティグループ ID を含める必要があります。

コマンド:

```
aws ec2 request-spot-instances --spot-price "0.050" --instance-count 1 --
type "one-time" --launch-specification file://specification.json
```

Specification.json:

```

{
  "ImageId": "ami-1a2b3c4d",
  "KeyName": "my-key-pair",
  "InstanceType": "m3.medium",
  "NetworkInterfaces": [
    {
      "DeviceIndex": 0,
      "SubnetId": "subnet-1a2b3c4d",
      "Groups": [ "sg-1a2b3c4d" ],
      "AssociatePublicIpAddress": true
    }
  ],

```

```
"IamInstanceProfile": {
  "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[RequestSpotInstances](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたサブネットに 1 回限りのスポットインスタンスをリクエストします。セキュリティグループは、指定されたサブネットを含む VPC 用に作成する必要があり、ネットワークインターフェイスを使用して ID で指定する必要があります。ネットワークインターフェイスの指定時、ネットワークインターフェイスを使用してサブネット ID を含める必要があります。

```
$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
```

出力:

```
ActualBlockHourlyPrice      :
AvailabilityZoneGroup       :
BlockDurationMinutes        : 0
CreateTime                  : 12/26/2015 7:44:10 AM
Fault                       :
InstanceId                  :
LaunchedAvailabilityZone    :
LaunchGroup                 :
LaunchSpecification         : Amazon.EC2.Model.LaunchSpecification
ProductDescription          : Linux/UNIX
SpotInstanceRequestId       : sir-12345678
```



```
SpotPrice      : 0.050000
State          : open
Status         : Amazon.EC2.Model.SpotInstanceStatus
Tags           : {}
Type           : one-time
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[RequestSpotInstances](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたサブネットに 1 回限りのスポットインスタンスをリクエストします。セキュリティグループは、指定されたサブネットを含む VPC 用に作成する必要があり、ネットワークインターフェイスを使用して ID で指定する必要があります。ネットワークインターフェイスの指定時、ネットワークインターフェイスを使用してサブネット ID を含める必要があります。

```
$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
```

出力:

```
ActualBlockHourlyPrice :
AvailabilityZoneGroup  :
BlockDurationMinutes   : 0
CreateTime             : 12/26/2015 7:44:10 AM
Fault                  :
InstanceId              :
LaunchedAvailabilityZone :
LaunchGroup            :
LaunchSpecification     : Amazon.EC2.Model.LaunchSpecification
ProductDescription      : Linux/UNIX
SpotInstanceRequestId  : sir-12345678
SpotPrice              : 0.050000
State                  : open
Status                 : Amazon.EC2.Model.SpotInstanceStatus
```

```
Tags          : {}  
Type          : one-time
```

- API の詳細については、AWS Tools for PowerShell 「コマンドレトリファレンス (V5)」の「[RequestSpotInstances](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **ResetImageAttribute** を使用する

次のサンプルコードは、ResetImageAttribute を使用する方法を説明しています。

CLI

AWS CLI

launchPermission 属性をリセットするには

この例では、指定された AMI の launchPermission 属性をデフォルト値にリセットします。デフォルトでは、AMI はプライベートです。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 reset-image-attribute --image-id ami-5731123e --  
attribute launchPermission
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ResetImageAttribute](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、「launchPermission」属性をデフォルト値にリセットします。デフォルトでは、AMI はプライベートです。

```
Reset-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ResetImageAttribute](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、「launchPermission」属性をデフォルト値にリセットします。デフォルトでは、AMI はプライベートです。

```
Reset-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[ResetImageAttribute](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **ResetInstanceAttribute** を使用する

次のサンプルコードは、ResetInstanceAttribute を使用方法を説明しています。

CLI

AWS CLI

sourceDestCheck 属性をリセットするには

この例では、指定されたインスタンスの sourceDestCheck 属性をリセットします。インスタンスは VPC にある必要があります。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 reset-instance-attribute --instance-id i-1234567890abcdef0 --  
attribute sourceDestCheck
```

カーネル属性をリセットするには

この例では、指定されたインスタンスの kernel 属性をリセットします。インスタンスは stopped の状態である必要があります。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 reset-instance-attribute --instance-id i-1234567890abcdef0 --  
attribute kernel
```

ramdisk 属性をリセットするには

この例では、指定されたインスタンスの ramdisk 属性をリセットします。インスタンスは stopped の状態である必要があります。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 reset-instance-attribute --instance-id i-1234567890abcdef0 --  
attribute ramdisk
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ResetInstanceAttribute](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたインスタンスの「sriovNetSupport」属性をリセットします。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

例 2: この例では、指定されたインスタンスの「ebsOptimized」属性をリセットします。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

例 3: この例では、指定されたインスタンスの「sourceDestCheck」属性をリセットします。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sourceDestCheck
```

例 4: この例では、指定されたインスタンスの「disableApiTermination」属性をリセットします。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute  
disableApiTermination
```

例 5: この例では、指定されたインスタンスの「instanceInitiatedShutdownBehavior」属性をリセットします。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute  
instanceInitiatedShutdownBehavior
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ResetInstanceAttribute](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたインスタンスの「sriovNetSupport」属性をリセットします。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

例 2: この例では、指定されたインスタンスの「ebsOptimized」属性をリセットします。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

例 3: この例では、指定されたインスタンスの「sourceDestCheck」属性をリセットします。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sourceDestCheck
```

例 4: この例では、指定されたインスタンスの「disableApiTermination」属性をリセットします。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute  
disableApiTermination
```

例 5: この例では、指定されたインスタンスの「instanceInitiatedShutdownBehavior」属性をリセットします。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute  
instanceInitiatedShutdownBehavior
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[ResetInstanceAttribute](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で `ResetNetworkInterfaceAttribute` を使用する

次のサンプルコードは、`ResetNetworkInterfaceAttribute` を使用方法を説明しています。

CLI

AWS CLI

ネットワークインターフェイス属性をリセットするには

次の `reset-network-interface-attribute` の例では、送信元/送信先チェック属性の値を `true` にリセットします。

```
aws ec2 reset-network-interface-attribute \
  --network-interface-id eni-686ea200 \
  --source-dest-check
```

このコマンドでは何も出力されません。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ResetNetworkInterfaceAttribute](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたネットワークインターフェイスの送信元/送信先チェックをリセットします。

```
Reset-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -
SourceDestCheck
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ResetNetworkInterfaceAttribute](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたネットワークインターフェイスの送信元/送信先チェックをリセットします。

```
Reset-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -  
SourceDestCheck
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[ResetNetworkInterfaceAttribute](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **ResetSnapshotAttribute** を使用する

次のサンプルコードは、ResetSnapshotAttribute を使用方法を説明しています。

CLI

AWS CLI

スナップショット属性をリセットするには

この例では、スナップショット snap-1234567890abcdef0 のボリューム作成のアクセス許可をリセットします。コマンドが成功した場合、出力は返りません。

コマンド:

```
aws ec2 reset-snapshot-attribute --snapshot-id snap-1234567890abcdef0 --  
attribute createVolumePermission
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[ResetSnapshotAttribute](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定したスナップショットに指定した属性を設定します。

```
Reset-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute  
CreateVolumePermission
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ResetSnapshotAttribute](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定したスナップショットに指定した属性を設定します。

```
Reset-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute  
CreateVolumePermission
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[ResetSnapshotAttribute](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **RevokeSecurityGroupEgress** を使用する

次のサンプルコードは、`RevokeSecurityGroupEgress` を使用する方法を説明しています。

CLI

AWS CLI

例 1: 特定のアドレス範囲へのアウトバウンドトラフィックを許可するルールを削除するには

次の `revoke-security-group-egress` の例では、TCP ポート 80 で指定されたアドレス範囲へのアクセスを付与するルールを削除します。

```
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}]}
```

このコマンドでは何も出力されません。

詳細については、「Amazon EC2 ユーザーガイド」の「[Security groups](#)」を参照してください。

例 2: 特定のセキュリティグループへのアウトバウンドトラフィックを許可するルールを削除するには

次の `revoke-security-group-egress` の例では、TCP ポート 80 で指定されたセキュリティグループへのアクセスを付与するルールを削除します。

```
aws ec2 revoke-security-group-egress \  
  --group-id sg-026c12253ce15eff7 \  
  --ip-permissions '[{"IpProtocol": "tcp", "FromPort": 443, "ToPort":  
443, "UserIdGroupPairs": [{"GroupId": "sg-06df23a01ff2df86d"}]}]'
```

このコマンドでは何も出力されません。

詳細については、「Amazon EC2 ユーザーガイド」の「[Security groups](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[RevokeSecurityGroupEgress](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、EC2-VPC の指定されたセキュリティグループのルールを削除します。これにより、TCP ポート 80 で指定された IP アドレス範囲へのアクセスが取り消されます。この例で使用する構文には、PowerShell バージョン 3 以降が必要です。

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

例 2: PowerShell バージョン 2 では、`New-Object` を使用して `IpPermission` オブジェクトを作成する必要があります。

```
$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
```

例 3: この例では、TCP ポート 80 で指定されたソースセキュリティグループへのアクセスを取り消します。

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission
@( @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; UserIdGroupPairs=$ug } )
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[RevokeSecurityGroupEgress](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、EC2-VPC の指定されたセキュリティグループのルールを削除します。これにより、TCP ポート 80 で指定された IP アドレス範囲へのアクセスが取り消されます。この例で使用する構文には、PowerShell バージョン 3 以降が必要です。

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

例 2: PowerShell バージョン 2 では、New-Object を使用して IpPermission オブジェクトを作成する必要があります。

```
$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
```

例 3: この例では、TCP ポート 80 で指定されたソースセキュリティグループへのアクセスを取り消します。

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission
@( @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; UserIdGroupPairs=$ug } )
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[RevokeSecurityGroupEgress](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で `RevokeSecurityGroupIngress` を使用する

次のサンプルコードは、`RevokeSecurityGroupIngress` を使用方法を説明しています。

CLI

AWS CLI

例 1: セキュリティグループからルールを削除するには

次の `revoke-security-group-ingress` の例では、デフォルトの VPC に指定されたセキュリティグループから、`203.0.113.0/24` アドレス範囲の TCP ポート 22 アクセスを削除します。

```
aws ec2 revoke-security-group-ingress \
  --group-name mySecurityGroup \
  --protocol tcp \
  --port 22 \
  --cidr 203.0.113.0/24
```

このコマンドが成功した場合、出力は生成されません。

詳細については、「Amazon EC2 ユーザーガイド」の「[Security groups](#)」を参照してください。

例 2: IP アクセス許可セットを使用してルールを削除するには

次の `revoke-security-group-ingress` の例では、`ip-permissions` パラメータを使用して、ICMP メッセージ Destination Unreachable: Fragmentation Needed and Don't Fragment was Set (タイプ 3、コード 4) を許可するインバウンドルールを削除します。

```
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}]
```

このコマンドが成功した場合、出力は生成されません。

詳細については、「Amazon EC2 ユーザーガイド」の「[Security groups](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[RevokeSecurityGroupIngress](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、EC2-VPC の指定されたセキュリティグループに対して、指定されたアドレス範囲から TCP ポート 22 へのアクセスを取り消します。セキュリティグループの名前ではなく、セキュリティグループの ID を使用して、EC2-VPC のセキュリティグループを識別する必要があります。この例で使用される構文には、PowerShell バージョン 3 以降が必要です。

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22";  
  IpRanges="203.0.113.0/24" }  
Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

例 2: PowerShell バージョン 2 では、New-Object を使用して IpPermission オブジェクトを作成する必要があります。

```
$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
```

例 3: この例では、EC2-Classic の指定されたセキュリティグループに対して、指定されたアドレス範囲から TCP ポート 22 へのアクセスを取り消します。この例で使用される構文には、PowerShell バージョン 3 以降が必要です。

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22";  
  IpRanges="203.0.113.0/24" }  
  
Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

例 4: PowerShell バージョン 2 では、New-Object を使用して IpPermission オブジェクトを作成する必要があります。

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 22
$ip.ToPort = 22
$ip.IpRanges.Add("203.0.113.0/24")

Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[RevokeSecurityGroupIngress](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、EC2-VPC の指定されたセキュリティグループに対して、指定されたアドレス範囲から TCP ポート 22 へのアクセスを取り消します。セキュリティグループの名前ではなく、セキュリティグループの ID を使用して、EC2-VPC のセキュリティグループを識別する必要があります。この例で使用される構文には、PowerShell バージョン 3 以降が必要です。

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22";
  IpRanges="203.0.113.0/24" }
Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

例 2: PowerShell バージョン 2 では、New-Object を使用して IpPermission オブジェクトを作成する必要があります。

```
$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
```

例 3: この例では、EC2-Classic の指定されたセキュリティグループに対して、指定されたアドレス範囲から TCP ポート 22 へのアクセスを取り消します。この例で使用される構文には、PowerShell バージョン 3 以降が必要です。

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22";  
  IpRanges="203.0.113.0/24" }
```

```
Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

例 4: PowerShell バージョン 2 では、New-Object を使用して IpPermission オブジェクトを作成する必要があります。

```
$ip = New-Object Amazon.EC2.Model.IpPermission  
$ip.IpProtocol = "tcp"  
$ip.FromPort = 22  
$ip.ToPort = 22  
$ip.IpRanges.Add("203.0.113.0/24")
```

```
Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[RevokeSecurityGroupIngress](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **RunInstances**で を使用する

次のサンプルコードは、RunInstances を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)
- [Amazon VPC の使用を開始する](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

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

```

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

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

```

- API の詳細については、AWS SDK for .NET API リファレンスの「[RunInstances](#)」を参照してください。

Bash

AWS CLI Bash スクリプトを使用する

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

#####
# 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
}

```

この例で使用されているユーティリティ関数。

```

#####
# 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
}

```

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[RunInstances](#)」を参照してください。

C++

SDK for C++

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

//! Launch an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
    \param instanceName: A name for the EC2 instance.
    \param amiId: An Amazon Machine Image (AMI) identifier.
    \param[out] instanceID: String to return the instance ID.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::EC2::runInstance(const Aws::String &instanceName,
                              const Aws::String &amiId,
                              Aws::String &instanceID,
                              const Aws::Client::ClientConfiguration
&clientConfiguration) {

```

```

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

    Aws::EC2::Model::RunInstancesRequest runRequest;
    runRequest.SetImageId(amiId);
    runRequest.SetInstanceType(Aws::EC2::Model::InstanceType::t1_micro);
    runRequest.SetMinCount(1);
    runRequest.SetMaxCount(1);

    Aws::EC2::Model::RunInstancesOutcome runOutcome = ec2Client.RunInstances(
        runRequest);
    if (!runOutcome.IsSuccess()) {
        std::cerr << "Failed to launch EC2 instance " << instanceName <<
            " based on ami " << amiId << ":" <<
            runOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    const Aws::Vector<Aws::EC2::Model::Instance> &instances =
runOutcome.GetResult().GetInstances();
    if (instances.empty()) {
        std::cerr << "Failed to launch EC2 instance " << instanceName <<
            " based on ami " << amiId << ":" <<
            runOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    instanceID = instances[0].GetInstanceId();

    return true;
}

```

- API の詳細については、AWS SDK for C++ API リファレンスの「[RunInstances](#)」を参照してください。

CLI

AWS CLI

例 1: インスタンスをデフォルトサブネット内で起動するには

次の run-instances の例では、現在のリージョンのデフォルトサブネットにタイプ t2.micro の単一のインスタンスを起動し、それをリージョンのデフォルト VPC のデフォル

トサブネットに関連付けます。SSH (Linux) または RDP (Windows) を使用してインスタンスに接続する予定がない場合、キーペアはオプションです。

```
aws ec2 run-instances \  
  --image-id ami-0abcdef1234567890 \  
  --instance-type t2.micro \  
  --key-name MyKeyPair
```

出力:

```
{  
  "Instances": [  
    {  
      "AmiLaunchIndex": 0,  
      "ImageId": "ami-0abcdef1234567890",  
      "InstanceId": "i-1231231230abcdef0",  
      "InstanceType": "t2.micro",  
      "KeyName": "MyKeyPair",  
      "LaunchTime": "2018-05-10T08:05:20.000Z",  
      "Monitoring": {  
        "State": "disabled"  
      },  
      "Placement": {  
        "AvailabilityZone": "us-east-2a",  
        "GroupName": "",  
        "Tenancy": "default"  
      },  
      "PrivateDnsName": "ip-10-0-0-157.us-east-2.compute.internal",  
      "PrivateIpAddress": "10.0.0.157",  
      "ProductCodes": [],  
      "PublicDnsName": "",  
      "State": {  
        "Code": 0,  
        "Name": "pending"  
      },  
      "StateTransitionReason": "",  
      "SubnetId": "subnet-04a636d18e83cfacb",  
      "VpcId": "vpc-1234567890abcdef0",  
      "Architecture": "x86_64",  
      "BlockDeviceMappings": [],  
      "ClientToken": "",  
      "EbsOptimized": false,  
      "Hypervisor": "xen",  
    }  
  ]  
}
```

```

    "NetworkInterfaces": [
      {
        "Attachment": {
          "AttachTime": "2018-05-10T08:05:20.000Z",
          "AttachmentId": "eni-attach-0e325c07e928a0405",
          "DeleteOnTermination": true,
          "DeviceIndex": 0,
          "Status": "attaching"
        },
        "Description": "",
        "Groups": [
          {
            "GroupName": "MySecurityGroup",
            "GroupId": "sg-0598c7d356eba48d7"
          }
        ],
        "Ipv6Addresses": [],
        "MacAddress": "0a:ab:58:e0:67:e2",
        "NetworkInterfaceId": "eni-0c0a29997760baee7",
        "OwnerId": "123456789012",
        "PrivateDnsName": "ip-10-0-0-157.us-east-2.compute.internal",
        "PrivateIpAddress": "10.0.0.157",
        "PrivateIpAddresses": [
          {
            "Primary": true,
            "PrivateDnsName": "ip-10-0-0-157.us-
east-2.compute.internal",
            "PrivateIpAddress": "10.0.0.157"
          }
        ],
        "SourceDestCheck": true,
        "Status": "in-use",
        "SubnetId": "subnet-04a636d18e83cfac",
        "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"
}

```

例 2: デフォルトではないサブネットでインスタンスを起動し、パブリック IP アドレスを追加するには

次の `run-instances` の例では、デフォルト以外のサブネットで起動するインスタンスのパブリック IP アドレスをリクエストします。インスタンスは指定されたセキュリティグループに関連付けられます。

```

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

```

`run-instances` の出力例については、例 1 を参照してください。

例 3: ボリュームを追加してインスタンスを起動するには

次の `run-instances` の例では、`mapping.json` で指定されたブロックデバイスマッピングを使用して、起動時に追加のボリュームをアタッチします。ブロックデバイスマッピングは、EBS ボリューム、インスタンスストアボリューム、あるいは EBS ボリュームとインスタンスストアボリュームの両方を指定できます。

```
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` の内容。この例では、サイズが 100 GiB である空の EBS ボリュームの `/dev/sdh` を追加します。

```
[  
  {  
    "DeviceName": "/dev/sdh",  
    "Ebs": {  
      "VolumeSize": 100  
    }  
  }  
]
```

`mapping.json` の内容。この例は、`ephemeral1` をインスタンスストアボリュームとして追加しています。

```
[  
  {  
    "DeviceName": "/dev/sdc",  
    "VirtualName": "ephemeral1"  
  }  
]
```

`run-instances` の出力例については、例 1 を参照してください。

ブロックデバイスマッピングの詳細については、「Amazon EC2 ユーザーガイド」の「[ブロックデバイスマッピング](#)」を参照してください。

例 4: インスタンスを起動し、作成時にタグを追加するには

次の run-instances の例では、キー webserver と値 production のタグをインスタンスに追加しています。さらに、cost-center キーと cc123 の値を持つタグを、作成された EBS ボリューム (この場合はルートボリューム) に適用します。

```
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}]'
```

run-instances の出力例については、例 1 を参照してください。

例 5: ユーザーデータを使用してインスタンスを起動するには

次の run-instances の例では、インスタンスの設定スクリプトを含む my_script.txt というファイルにユーザーデータを渡します。このスクリプトは起動時に実行されます。

```
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
```

run-instances の出力例については、例 1 を参照してください。

インスタンスユーザーデータの詳細については、「Amazon EC2 ユーザーガイド」の「[インスタンスユーザーデータの使用](#)」を参照してください。

例 6: バーストパフォーマンスインスタンスを起動するには

次の run-instances の例では、unlimited クレジットオプションを使用して t2.micro インスタンスを起動しています。T2 インスタンスを起動する際に --credit-specification

を指定しない場合、デフォルトは standard クレジットオプションです。T3 インスタンスを起動する際、デフォルトは unlimited クレジットオプションです。

```
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
```

run-instances の出力例については、例 1 を参照してください。

バーストパフォーマンスインスタンスの詳細については、「Amazon EC2 ユーザーガイド」の「[バーストパフォーマンスインスタンス](#)」を参照してください。

- API の詳細については、「AWS CLI Command Reference」の「[RunInstances](#)」を参照してください。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/**  
 * Runs an EC2 instance asynchronously.  
 *  
 * @param instanceType The instance type to use for the EC2 instance.  
 * @param keyName The name of the key pair to associate with the EC2  
 instance.  
 * @param groupName The name of the security group to associate with the EC2  
 instance.  
 * @param amiId The ID of the Amazon Machine Image (AMI) to use for the EC2  
 instance.
```

```
    * @return A {@link CompletableFuture} that completes with the ID of the
    started EC2 instance.
    * @throws RuntimeException If there is an error running the EC2 instance.
    */
    public CompletableFuture<String> runInstanceAsync(String instanceType, String
keyName, String groupName, String amiId) {
        RunInstancesRequest runRequest = RunInstancesRequest.builder()
            .instanceType(instanceType)
            .keyName(keyName)
            .securityGroups(groupName)
            .maxCount(1)
            .minCount(1)
            .imageId(amiId)
            .build();

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

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[RunInstances](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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.
    // https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/finding-an-ami.html
    ImageId: imageId,
```

```
// An instance type describing the resources provided to your instance. There
// are multiple
// ways to search for instance types. For more information see:
// https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-
// discovery.html
InstanceType: instanceType,
// Availability Zones have capacity limitations that may impact your ability
// to launch instances.
// The `RunInstances` operation will only succeed if it can allocate at least
// the `MinCount` of instances.
// However, EC2 will attempt to launch up to the `MaxCount` of instances,
// even if the full request cannot be satisfied.
// If you need a specific number of instances, use `MinCount` and `MaxCount`
// set to the same value.
// If you want to launch up to a certain number of instances, use `MaxCount`
// and let EC2 provision as many as possible.
// If you require a minimum number of instances, but do not want to exceed a
// maximum, use both `MinCount` and `MaxCount`.
MinCount: minCount,
MaxCount: maxCount,
});

try {
  const { Instances } = await client.send(command);
  const instanceList = Instances.map(
    (instance) => `• ${instance.InstanceId}`,
  ).join("\n");
  console.log(`Launched instances:\n${instanceList}`);
} catch (caught) {
  if (caught instanceof Error && caught.name === "ResourceCountExceeded") {
    console.warn(`${caught.message}`);
  } else {
    throw caught;
  }
}
};
```

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の「[RunInstances](#)」を参照してください。

Kotlin

SDK for Kotlin

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[RunInstances](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、EC2-Classic またはデフォルトの VPC で指定された AMI の単一のインスタンスを起動します。

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -InstanceType
m3.medium -KeyName my-key-pair -SecurityGroup my-security-group
```

例 2: この例では、VPC で指定された AMI の単一のインスタンスを起動します。

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -SubnetId
subnet-12345678 -InstanceType t2.micro -KeyName my-key-pair -SecurityGroupId
sg-12345678
```

例 3: EBS ボリュームまたはインスタンスストアボリュームを追加するには、ブロックデバイスマッピングを定義してコマンドに追加します。この例では、インスタンスストアボリュームとして追加します。

```
$bdm = New-Object Amazon.EC2.Model.BlockDeviceMapping
$bdm.VirtualName = "ephemeral0"
$bdm.DeviceName = "/dev/sdf"

New-EC2Instance -ImageId ami-12345678 -BlockDeviceMapping $bdm ...
```

例 4: 現在の Windows AMI のいずれかを指定するには、Get-EC2ImageByName を使用して AMI ID を取得します。この例では、Windows Server 2016 の現在のベース AMI からインスタンスを起動します。

```
$ami = Get-EC2ImageByName WINDOWS_2016_BASE

New-EC2Instance -ImageId $ami.ImageId ...
```

例 5: 指定された専有ホスト環境でインスタンスを起動します。


```
New-EC2Instance -ImageId ami-1a2b3c4d -InstanceType m4.large -KeyName my-key-pair  
-SecurityGroupId sg-1a2b3c4d -AvailabilityZone us-west-1a -Tenancy host -HostID  
h-1a2b3c4d5e6f1a2b3
```

例 6: このリクエストは 2 つのインスタンスを起動し、ウェブサーバーのキーと本番稼働用の値を持つタグをインスタンスに適用します。リクエストは、作成されたボリューム (この場合は各インスタンスのルートボリューム) に、コストセンターのキーと cc123 の値を持つタグも適用します。

```
$tag1 = @{ Key="webserver"; Value="production" }  
$tag2 = @{ Key="cost-center"; Value="cc123" }  
  
$tagspec1 = new-object Amazon.EC2.Model.TagSpecification  
$tagspec1.ResourceType = "instance"  
$tagspec1.Tags.Add($tag1)  
  
$tagspec2 = new-object Amazon.EC2.Model.TagSpecification  
$tagspec2.ResourceType = "volume"  
$tagspec2.Tags.Add($tag2)  
  
New-EC2Instance -ImageId "ami-1a2b3c4d" -KeyName "my-key-pair" -MaxCount 2 -  
InstanceType "t2.large" -SubnetId "subnet-1a2b3c4d" -TagSpecification $tagspec1,  
$tagspec2
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[RunInstances](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、EC2-Classic またはデフォルトの VPC で指定された AMI の単一のインスタンスを起動します。

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -InstanceType  
m3.medium -KeyName my-key-pair -SecurityGroup my-security-group
```

例 2: この例では、VPC で指定された AMI の単一のインスタンスを起動します。

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -SubnetId  
subnet-12345678 -InstanceType t2.micro -KeyName my-key-pair -SecurityGroupId  
sg-12345678
```

例 3: EBS ボリュームまたはインスタンスストアボリュームを追加するには、ブロックデバイスマッピングを定義してコマンドに追加します。この例では、インスタンスストアボリュームとして追加します。

```
$bdm = New-Object Amazon.EC2.Model.BlockDeviceMapping
$bdm.VirtualName = "ephemeral0"
$bdm.DeviceName = "/dev/sdf"

New-EC2Instance -ImageId ami-12345678 -BlockDeviceMapping $bdm ...
```

例 4: 現在の Windows AMI のいずれかを指定するには、Get-SSMLatestEC2Image を使用して AMI ID を取得します。この例では、Windows Server 2016 の現在のベース AMI からインスタンスを起動します。

```
$ami = (Get-SSMLatestEC2Image -Path 'ami-windows-latest' -ImageName
'WINDOWS*2016*English*Core*BASE').Value

New-EC2Instance -ImageId $ami ...
```

例 5: 指定された専用ホスト環境でインスタンスを起動します。

```
New-EC2Instance -ImageId ami-1a2b3c4d -InstanceType m4.large -KeyName my-key-pair
-SecurityGroupId sg-1a2b3c4d -AvailabilityZone us-west-1a -Tenancy host -HostID
h-1a2b3c4d5e6f1a2b3
```

例 6: このリクエストは 2 つのインスタンスを起動し、ウェブサーバーのキーと本番稼働用の値を持つタグをインスタンスに適用します。リクエストは、作成されたボリューム (この場合は各インスタンスのルートボリューム) に、コストセンターのキーと cc123 の値を持つタグも適用します。

```
$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
```

例 7: この例では、インスタンスを実際に作成せずに DryRun パラメータを使用して EC2 インスタンスを起動するためのアクセス許可を検証します。注: 成功すると例外がスローされますが、これは正常な動作です。

```
New-EC2Instance -ImageId 'ami-12345678' -InstanceType 't2.micro' -KeyName 'my-
key-pair' -Region 'us-west-2' -DryRun $true
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[RunInstances](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
    using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
        instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
        level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
        objects. These are high-level objects that
```

```

        wrap instance actions.

    """
    self.ec2_client = ec2_client
    self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
        EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def create(
        self,
        image_id: str,
        instance_type: str,
        key_pair_name: str,
        security_group_ids: Optional[List[str]] = None,
    ) -> List[Dict[str, Any]]:
        """
        Creates a new EC2 instance in the default VPC of the current account.

        The instance starts immediately after it is created.

        :param image_id: The ID of the Amazon Machine Image (AMI) to use for the
        instance.
        :param instance_type: The type of instance to create, such as 't2.micro'.
        :param key_pair_name: The name of the key pair to use for SSH access.
        :param security_group_ids: A list of security group IDs to associate with
        the instance.

        If not specified, the default security group
        of the VPC is used.

        :return: A list of dictionaries representing Boto3 Instance objects
        representing the newly created instances.
        """
        try:
            instance_params = {
                "ImageId": image_id,
                "InstanceType": instance_type,

```

```

        "KeyName": key_pair_name,
    }
    if security_group_ids is not None:
        instance_params["SecurityGroupIds"] = security_group_ids

    response = self.ec2_client.run_instances(
        **instance_params, MinCount=1, MaxCount=1
    )
    instance = response["Instances"][0]
    self.instances.append(instance)
    waiter = self.ec2_client.get_waiter("instance_running")
    waiter.wait(InstanceIds=[instance["InstanceId"]])
except ClientError as err:
    params_str = "\n\t".join(
        f"{key}: {value}" for key, value in instance_params.items()
    )
    logger.error(
        f"Failed to complete instance creation request.\nRequest details:
{params_str}"
    )
    error_code = err.response["Error"]["Code"]
    if error_code == "InstanceLimitExceeded":
        logger.error(
            (
                f"Insufficient capacity for instance type
'{instance_type}'. "
                "Terminate unused instances or contact AWS Support for a
limit increase."
            )
        )
    if error_code == "InsufficientInstanceCapacity":
        logger.error(
            (
                f"Insufficient capacity for instance type
'{instance_type}'. "
                "Select a different instance type or launch in a
different availability zone."
            )
        )
    raise
return self.instances

```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[RunInstances](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn create_instance<'a>(
    &self,
    image_id: &'a str,
    instance_type: InstanceType,
    key_pair: &'a KeyPairInfo,
    security_groups: Vec<&'a SecurityGroup>,
) -> Result<String, EC2Error> {
    let run_instances = self
        .client
        .run_instances()
        .image_id(image_id)
        .instance_type(instance_type)
        .key_name(
            key_pair
                .key_name()
                .ok_or_else(|| EC2Error::new("Missing key name when launching
instance"))?,
        )
        .set_security_group_ids(Some(
            security_groups
                .iter()
                .filter_map(|sg| sg.group_id.clone())
                .collect(),
        ))
        .min_count(1)
        .max_count(1)
        .send()
        .await?;
```

```
if run_instances.instances().is_empty() {
    return Err(EC2Error::new("Failed to create instance"));
}

let instance_id = run_instances.instances()[0].instance_id().unwrap();
let response = self
    .client
    .create_tags()
    .resources(instance_id)
    .tags(
        Tag::builder()
            .key("Name")
            .value("From SDK Examples")
            .build(),
    )
    .send()
    .await;

match response {
    Ok(_) => tracing::info!("Created {instance_id} and applied tags."),
    Err(err) => {
        tracing::info!("Error applying tags to {instance_id}: {err:?}");
        return Err(err.into());
    }
}

tracing::info!("Instance is created.");

Ok(instance_id.to_string())
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[RunInstances](#)」を参照してください。

SAP ABAP

SDK for SAP ABAP

 Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
" 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(                                " oo_result
is returned for testing purposes. "
    iv_imageid = iv_ami_id
    iv_instancetype = 't3.micro'
    iv_maxcount = 1
    iv_mincount = 1
    it_tagsspecifications = lt_tagsspecifications
    iv_subnetid = iv_subnet_id ).
  MESSAGE 'EC2 instance created.' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
  DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
  MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「[RunInstances](#)」を参照してください。

Swift

SDK for Swift

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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
    }
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[RunInstances](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **RunScheduledInstances** を使用する

次のサンプルコードは、RunScheduledInstances を使用する方法を説明しています。

CLI

AWS CLI

スケジュールされたインスタンスを起動するには

この例では、VPC で指定されたスケジュールされたインスタンスを起動します。

コマンド:

```
aws ec2 run-scheduled-instances --scheduled-instance-
id sci-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"
  }
}
```

出力:

```
{
  "InstanceIdSet": [
    "i-1234567890abcdef0"
  ]
}
```

この例では、EC2-Classic で指定されたスケジュールされたインスタンスを起動します。

コマンド:

```
aws ec2 run-scheduled-instances --scheduled-instance-
id sci-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"
  }
}
```

出力:

```
{
  "InstanceIdSet": [
    "i-1234567890abcdef0"
  ]
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[RunScheduledInstances](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定のスケジュールされたインスタンスを起動します。

```
New-EC2ScheduledInstance -ScheduledInstanceId
sci-1234-1234-1234-1234-123456789012 -InstanceCount 1 `
-IamInstanceProfile_Name my-iam-role `
-LaunchSpecification_ImageId ami-12345678 `
-LaunchSpecification_InstanceType c4.large `
-LaunchSpecification_SubnetId subnet-12345678`
-LaunchSpecification_SecurityGroupId sg-12345678
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[RunScheduledInstances](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定のスケジュールされたインスタンスを起動します。

```
New-EC2ScheduledInstance -ScheduledInstanceId
sci-1234-1234-1234-1234-123456789012 -InstanceCount 1 `
-IamInstanceProfile_Name my-iam-role `
-LaunchSpecification_ImageId ami-12345678 `
-LaunchSpecification_InstanceType c4.large `
-LaunchSpecification_SubnetId subnet-12345678`
-LaunchSpecification_SecurityGroupId sg-12345678
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[RunScheduledInstances](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **StartInstances**で を使用する

次のサンプルコードは、StartInstances を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

        await _amazonEC2.StartInstancesAsync(request);

        Console.WriteLine("Waiting for instance to start. ");
    }
}
```

```

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

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

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

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

        // Check for the desired state.
        var response = await _amazonEC2.DescribeInstancesAsync(request);

```

```

        var instance = response.Reservations[0].Instances[0];
        hasState = instance.State.Name == stateName;
        Console.WriteLine(". ");
    } while (!hasState);

    return hasState;
}

```

- APIの詳細については、AWS SDK for .NET API リファレンスの「[StartInstances](#)」を参照してください。

Bash

AWS CLI Bash スクリプトを使用する

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

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

```

```

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

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

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

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

return 0
}

```


この例で使用されているユーティリティ関数。

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

- API の詳細については、「AWS CLI コマンドリファレンス」の「[StartInstances](#)」を参照してください。

C++

SDK for C++

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
#!/ Start an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
    \param instanceID: An EC2 instance ID.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::EC2::startInstance(const Aws::String &instanceId,
                                const Aws::Client::ClientConfiguration
                                &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::StartInstancesRequest startRequest;
    startRequest.AddInstanceIds(instanceId);
    startRequest.SetDryRun(true);

    Aws::EC2::Model::StartInstancesOutcome dryRunOutcome =
    ec2Client.StartInstances(startRequest);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
            << "Failed dry run to start instance. A dry run should trigger an
error."
            << std::endl;
        return false;
    }
```

```

    } else if (dryRunOutcome.GetError().GetErrorType() !=
               Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to start instance " << instanceId << ": "
                  << dryRunOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    startRequest.SetDryRun(false);
    Aws::EC2::Model::StartInstancesOutcome startInstancesOutcome =
    ec2Client.StartInstances(startRequest);

    if (!startInstancesOutcome.IsSuccess()) {
        std::cout << "Failed to start instance " << instanceId << ": " <<
                  startInstancesOutcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully started instance " << instanceId <<
                  std::endl;
    }

    return startInstancesOutcome.IsSuccess();
}

```

- API の詳細については、AWS SDK for C++ API リファレンスの「[StartInstances](#)」を参照してください。

CLI

AWS CLI

Amazon EC2 インスタンスを開始するには

この例では、指定された Amazon EBS-backed インスタンスを開始します。

コマンド:

```
aws ec2 start-instances --instance-ids i-1234567890abcdef0
```

出力:

```
{
  "StartingInstances": [
```

```
{
  "InstanceId": "i-1234567890abcdef0",
  "CurrentState": {
    "Code": 0,
    "Name": "pending"
  },
  "PreviousState": {
    "Code": 80,
    "Name": "stopped"
  }
}
```

詳細については、「Amazon Elastic Compute Cloud ユーザーガイド」の「インスタンスの停止と起動」を参照してください。

- API の詳細については、「AWS CLI Command Reference」の「[StartInstances](#)」を参照してください。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

```
        .build();

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

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

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

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[StartInstances](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import { EC2Client, StartInstancesCommand } from "@aws-sdk/client-ec2";
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

/**
 * Starts an Amazon EBS-backed instance that you've previously stopped.
 * @param {{ instanceIds }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new StartInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { StartingInstances } = await client.send(command);
    const instanceIdList = StartingInstances.map(
      (instance) => ` • ${instance.InstanceId}`,
    );
    console.log("Starting instances:");
    console.log(instanceIdList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の「[StartInstances](#)」を参照してください。

Kotlin

SDK for Kotlin

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[StartInstances](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたインスタンスを開始します。

```
Start-EC2Instance -InstanceId i-12345678
```

出力:

CurrentState	InstanceId	PreviousState
-----	-----	-----
Amazon.EC2.Model.InstanceState	i-12345678	Amazon.EC2.Model.InstanceState

例 2: この例では、指定されたインスタンスを開始します。

```
@("i-12345678", "i-76543210") | Start-EC2Instance
```

例 3: この例では、現在停止しているインスタンスのセットを開始します。Get-EC2Instance によって返されるインスタンスオブジェクトは、Start-EC2Instance にパイプされます。この例で使用される構文には、PowerShell バージョン 3 以降が必要です。

```
(Get-EC2Instance -Filter @{ Name="instance-state-name";  
  Values="stopped"}).Instances | Start-EC2Instance
```

例 4: PowerShell バージョン 2 では、New-Object を使用してフィルターパラメータのフィルターを作成する必要があります。

```
$filter = New-Object Amazon.EC2.Model.Filter  
$filter.Name = "instance-state-name"  
$filter.Values = "stopped"  
  
(Get-EC2Instance -Filter $filter).Instances | Start-EC2Instance
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[StartInstances](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたインスタンスを開始します。

```
Start-EC2Instance -InstanceId i-12345678
```

出力:

CurrentState	InstanceId	PreviousState
-----	-----	-----
Amazon.EC2.Model.InstanceState	i-12345678	Amazon.EC2.Model.InstanceState

例 2: この例では、指定されたインスタンスを開始します。


```
@("i-12345678", "i-76543210") | Start-EC2Instance
```

例 3: この例では、現在停止しているインスタンスのセットを開始します。Get-EC2Instance によって返されるインスタンスオブジェクトは、Start-EC2Instance にパイプされます。この例で使用される構文には、PowerShell バージョン 3 以降が必要です。

```
(Get-EC2Instance -Filter @{ Name="instance-state-name";  
  Values="stopped"}).Instances | Start-EC2Instance
```

例 4: PowerShell バージョン 2 では、New-Object を使用してフィルターパラメータのフィルターを作成する必要があります。

```
$filter = New-Object Amazon.EC2.Model.Filter  
$filter.Name = "instance-state-name"  
$filter.Values = "stopped"  
  
(Get-EC2Instance -Filter $filter).Instances | Start-EC2Instance
```

例 5: この例では、インスタンスを実際を開始せずに DryRun パラメータを使用して EC2 インスタンスを開始するためのアクセス許可を検証します。注: 成功すると例外がスローされますが、これは正常な動作です。

```
Start-EC2Instance -InstanceId 'i-0abcdef123456' -Region 'us-west-1' -DryRun $true
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[StartInstances](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
class EC2InstanceWrapper:
```

```

    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
    using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
        instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
objects. These are high-level objects that
                           wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def start(self) -> Optional[Dict[str, Any]]:
        """
        Starts instances and waits for them to be in a running state.

        :return: The response to the start request.
        """
        if not self.instances:
            logger.info("No instances to start.")
            return None

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:

```

```
        start_response =
self.ec2_client.start_instances(InstanceIds=instance_ids)
        waiter = self.ec2_client.get_waiter("instance_running")
        waiter.wait(InstanceIds=instance_ids)
        return start_response
    except ClientError as err:
        logger.error(
            f"Failed to start instance(s): {' '.join(map(str,
instance_ids))}"
        )
        error_code = err.response["Error"]["Code"]
        if error_code == "IncorrectInstanceState":
            logger.error(
                "Couldn't start instance(s) because they are in an incorrect
state. "
                "Ensure the instances are in a stopped state before starting
them."
            )
        raise
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[StartInstances](#)」を参照してください。

Ruby

SDK for Ruby

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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
end

ec2_client.start_instances(instance_ids: [instance_id])
ec2_client.wait_until(:instance_running, instance_ids: [instance_id])
puts 'Instance started.'
true
rescue StandardError => e
  puts "Error starting instance: #{e.message}"
  false
end

# Example usage:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.

```

```
if ARGV[0] == '--help' || ARGV[0] == '-h'
  puts 'Usage:  ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
    'INSTANCE_ID REGION '
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  puts 'Example: ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
    'i-123abc us-west-2'
  exit 1
# If no values are specified at the command prompt, use these default values.
# Replace us-west-2 with the AWS Region you're using for Amazon EC2.
elsif ARGV.count.zero?
  instance_id = 'i-123abc'
  region = 'us-west-2'
# Otherwise, use the values as specified at the command prompt.
else
  instance_id = ARGV[0]
  region = ARGV[1]
end

ec2_client = Aws::EC2::Client.new(region: region)

puts "Attempting to start instance '#{instance_id}' " \
  '(this might take a few minutes)... '
return if instance_started?(ec2_client, instance_id)

puts 'Could not start instance.'
end

run_me if $PROGRAM_NAME == __FILE__
```

- APIの詳細については、AWS SDK for Ruby API リファレンスの「[StartInstances](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

インスタンス ID で EC2 インスタンスを起動します。

```
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(())
}
```

Waiters API を使用して、インスタンスが準備完了状態およびステータス OK 状態になるまで待ちます。Waiters API を使用するには、rust ファイルで `use aws\_sdk\_ec2::client::Waiters` を使用する必要があります。

```
/// 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(())
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[StartInstances](#)」を参照してください。

SAP ABAP

SDK for SAP ABAP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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.

```

- APIの詳細については、「AWS SDK for SAP ABAP API リファレンス」の「[StartInstances](#)」を参照してください。

Swift

SDK for Swift

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

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  
    }  
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[StartInstances](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **StopInstances**で を使用する

次のサンプルコードは、StopInstances を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

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

        Console.WriteLine("\nThe instance has stopped.");
    }
    catch (AmazonEC2Exception ec2Exception)
    {
    }
}
```

```

        if (ec2Exception.ErrorCode == "InvalidInstanceId")
        {
            _logger.LogError(
                $"InstanceId is invalid, unable to stop.
{ec2Exception.Message}");
        }

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

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

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

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

```

```
        return hasState;
    }
}
```

- API の詳細については、AWS SDK for .NET API リファレンスの「[StopInstances](#)」を参照してください。

Bash

AWS CLI Bash スクリプトを使用する

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

    # bashsupport disable=BP5008
    function usage() {
        echo "function ec2_stop_instances"
        echo "Stops one or more Amazon Elastic Compute Cloud (Amazon EC2) instances."
    }
}
```

```
    echo "  -i instance_id - The ID(s) of the instance(s) to stop (comma-
separated).\"
    echo "  -h - Display help.\"
    echo \"\"
}

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

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

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

return 0
}
```

この例で使用されているユーティリティ関数。

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

- APIの詳細については、「AWS CLI コマンドリファレンス」の「[StopInstances](#)」を参照してください。

C++

SDK for C++

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
#!/ Stop an EC2 instance.
/*!
 \param instanceID: An EC2 instance ID.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::EC2::stopInstance(const Aws::String &instanceId,
                               const Aws::Client::ClientConfiguration
                               &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::StopInstancesRequest request;
    request.AddInstanceIds(instanceId);
    request.SetDryRun(true);

    Aws::EC2::Model::StopInstancesOutcome dryRunOutcome =
    ec2Client.StopInstances(request);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
            << "Failed dry run to stop instance. A dry run should trigger an
error."
            << std::endl;
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType() !=
        Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to stop instance " << instanceId << ": "
            << dryRunOutcome.GetError().GetMessage() << std::endl;
        return false;
    }
}
```

```
request.SetDryRun(false);
Aws::EC2::Model::StopInstancesOutcome outcome =
ec2Client.StopInstances(request);
if (!outcome.IsSuccess()) {
    std::cout << "Failed to stop instance " << instanceId << ": " <<
        outcome.GetError().GetMessage() << std::endl;
} else {
    std::cout << "Successfully stopped instance " << instanceId <<
        std::endl;
}

return outcome.IsSuccess();
}

void PrintUsage() {
    std::cout << "Usage: run_start_stop_instance <instance_id> <start|stop>" <<
        std::endl;
}
```

- APIの詳細については、AWS SDK for C++ API リファレンスの「[StopInstances](#)」を参照してください。

CLI

AWS CLI

例 1: Amazon EC2 インスタンスを停止するには

次の stop-instances の例では、Amazon EBS-backed インスタンスを停止します。

```
aws ec2 stop-instances \
    --instance-ids i-1234567890abcdef0
```

出力:

```
{
  "StoppingInstances": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "CurrentState": {
```



```
        "Code": 64,\n        "Name": "stopping"\n    },\n    "PreviousState": {\n        "Code": 16,\n        "Name": "running"\n    }\n  }\n]\n}
```

詳細については、「Amazon Elastic Compute Cloud ユーザーガイド」の「[インスタンスの停止と起動](#)」を参照してください。

例 2: Amazon EC2 インスタンスを休止するには

次の `stop-instances` の例では、休止が有効で、休止の前提条件を満たしている場合に Amazon EBS-backed インスタンスを休止します。インスタンスが休止状態になると、インスタンスは停止されます。

```
aws ec2 stop-instances \n  --instance-ids i-1234567890abcdef0 \n  --hibernate
```

出力:

```
{\n  "StoppingInstances": [\n    {\n      "CurrentState": {\n        "Code": 64,\n        "Name": "stopping"\n      },\n      "InstanceId": "i-1234567890abcdef0",\n      "PreviousState": {\n        "Code": 16,\n        "Name": "running"\n      }\n    }\n  ]\n}
```

詳細については、「Amazon Elastic Compute Cloud ユーザーガイド」の「[オンデマンド Linux インスタンスの休止方法](#)」を参照してください。

- API の詳細については、「AWS CLI Command Reference」の「[StopInstances](#)」を参照してください。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

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

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

    CompletableFuture<Void> resultFuture = new CompletableFuture<>();
    logger.info("Stopping instance " + instanceId + " and waiting for it to
stop.");
    getAsyncClient().stopInstances(stopRequest)
```

```

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

    return resultFuture;
}

```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[StopInstances](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

import { EC2Client, StopInstancesCommand } from "@aws-sdk/client-ec2";
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

```

```
/**
 * Stop one or more EC2 instances.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new StopInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { StoppingInstances } = await client.send(command);
    const instanceIdList = StoppingInstances.map(
      (instance) => ` • ${instance.InstanceId}`,
    );
    console.log("Stopping instances:");
    console.log(instanceIdList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[StopInstances](#)」を参照してください。

Kotlin

SDK for Kotlin

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[StopInstances](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたインスタンスを停止します。

```
Stop-EC2Instance -InstanceId i-12345678
```

出力:

CurrentState	InstanceId	PreviousState
-----	-----	-----
Amazon.EC2.Model.InstanceState	i-12345678	Amazon.EC2.Model.InstanceState

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[StopInstances](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたインスタンスを停止します。

```
Stop-EC2Instance -InstanceId i-12345678
```

出力:

CurrentState	InstanceId	PreviousState
-----	-----	-----
Amazon.EC2.Model.InstanceState	i-12345678	Amazon.EC2.Model.InstanceState

- APIの詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[StopInstances](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
    using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
        instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
        level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
        objects. These are high-level objects that
                           wrap instance actions.
        """
        self.ec2_client = ec2_client
```

```

        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
        EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def stop(self) -> Optional[Dict[str, Any]]:
        """
        Stops instances and waits for them to be in a stopped state.

        :return: The response to the stop request, or None if there are no
        instances to stop.
        """
        if not self.instances:
            logger.info("No instances to stop.")
            return None

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            # Attempt to stop the instances
            stop_response =
self.ec2_client.stop_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_stopped")
            waiter.wait(InstanceIds=instance_ids)
        except ClientError as err:
            logger.error(
                f"Failed to stop instance(s): {' '.join(map(str, instance_ids))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "IncorrectInstanceState":
                logger.error(
                    "Couldn't stop instance(s) because they are in an incorrect
state. "
                    "Ensure the instances are in a running state before stopping
them."
                )

```

```
        raise
    return stop_response
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[StopInstances](#)」を参照してください。

Ruby

SDK for Ruby

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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

  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)...'\n
return if instance_stopped?(ec2_client, instance_id)\n\nputs 'Could not stop instance.'\nend\n\nrun_me if $PROGRAM_NAME == __FILE__
```

- APIの詳細については、AWS SDK for Ruby API リファレンスの「[StopInstances](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
pub async fn stop_instance(&self, instance_id: &str) -> Result<(), EC2Error>\n{\n    tracing::info!("Stopping instance {instance_id}");\n\n    self.client\n        .stop_instances()\n        .instance_ids(instance_id)\n        .send()\n        .await?;\n\n    self.wait_for_instance_stopped(instance_id, None).await?;\n\n    tracing::info!("Stopped instance.");\n\n    Ok(())\n}
```

Writers API を使用して、インスタンスが停止状態になるまで待ちます。Writers API を使用するには、rust ファイルで `use aws\_sdk\_ec2::client::Waiters` を使用する必要があります。

```
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(())
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[StopInstances](#)」を参照してください。

SAP ABAP

SDK for SAP ABAP

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
DATA lt_instance_ids TYPE /aws1/
cl_ec2instidstringlist_w=>tt_instanceidstringlist.
APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
lt_instance_ids.

"Perform dry run"
```

```

TRY.
    " DryRun is set to true. This checks for the required permissions to stop
    the instance without actually making the request. "
    lo_ec2->stopinstances(
        it_instanceids = lt_instance_ids
        iv_dryrun = abap_true ).
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    " If the error code returned is `DryRunOperation`, then you have the
    required permissions to stop this instance. "
    IF lo_exception->av_err_code = 'DryRunOperation'.
        MESSAGE 'Dry run to stop instance completed.' TYPE 'I'.
        " DryRun is set to false to stop instance. "
        oo_result = lo_ec2->stopinstances(          " oo_result is returned
        for testing purposes. "
            it_instanceids = lt_instance_ids
            iv_dryrun = abap_false ).
        MESSAGE 'Successfully stopped the EC2 instance.' TYPE 'I'.
        " If the error code returned is `UnauthorizedOperation`, then you don't
        have the required permissions to stop this instance. "
        ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
            MESSAGE 'Dry run to stop instance failed. User does not have
            permissions to stop the instance.' TYPE 'E'.
        ELSE.
            DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
            >av_err_msg }|.
            MESSAGE lv_error TYPE 'E'.
        ENDIF.
    ENDTRY.

```

- APIの詳細については、「AWS SDK for SAP ABAP API リファレンス」の「[StopInstances](#)」を参照してください。

Swift

SDK for Swift

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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
    }
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[StopInstances](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **TerminateInstances**で を使用する

次のサンプルコードは、`TerminateInstances` を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)
- [Amazon VPC の使用を開始する](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

- API の詳細については、AWS SDK for .NET API リファレンスの「[TerminateInstances](#)」を参照してください。

Bash

AWS CLI Bash スクリプトを使用する

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。


```
#####
# 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
}

```

この例で使用されているユーティリティ関数。

```

#####
# 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
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[TerminateInstances](#)」を参照してください。

C++

SDK for C++

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
#!/ Terminate an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
```

```

\param instanceID: An EC2 instance ID.
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::EC2::terminateInstances(const Aws::String &instanceID,
                                     const Aws::Client::ClientConfiguration
                                     &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::TerminateInstancesRequest request;
    request.SetInstanceIds({instanceID});

    Aws::EC2::Model::TerminateInstancesOutcome outcome =
        ec2Client.TerminateInstances(request);
    if (outcome.IsSuccess()) {
        std::cout << "Ec2 instance '" << instanceID <<
            "' was terminated." << std::endl;
    } else {
        std::cerr << "Failed to terminate ec2 instance '" << instanceID <<
            "', " <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    return outcome.IsSuccess();
}

```

- APIの詳細については、AWS SDK for C++ API リファレンスの「[TerminateInstances](#)」を参照してください。

CLI

AWS CLI

Amazon EC2 インスタンスを終了するには

この例では、指定されたインスタンスを終了します。

コマンド:

```
aws ec2 terminate-instances --instance-ids i-1234567890abcdef0
```

出力:

```
{
  "TerminatingInstances": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "CurrentState": {
        "Code": 32,
        "Name": "shutting-down"
      },
      "PreviousState": {
        "Code": 16,
        "Name": "running"
      }
    }
  ]
}
```

詳細については、「AWS コマンドラインインターフェイスユーザーガイド」で Amazon EC2 インスタンスの使用方法を参照してください。

- API の詳細については、「AWS CLI Command Reference」の「[TerminateInstances](#)」を参照してください。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
/**
 * Terminates an EC2 instance asynchronously and waits for it to reach the
 * terminated state.
 *
 * @param instanceId the ID of the EC2 instance to terminate
 * @return a {@link CompletableFuture} that completes when the instance has
 * been terminated
 */
```

```
* @throws RuntimeException if there is no response from the AWS SDK or if
there is a failure during the termination process
*/
public CompletableFuture<Object> terminateEC2Async(String instanceId) {
    TerminateInstancesRequest terminateRequest =
    TerminateInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

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

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[TerminateInstances](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
import { EC2Client, TerminateInstancesCommand } from "@aws-sdk/client-ec2";
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

/**
 * Terminate one or more EC2 instances.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new TerminateInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { TerminatingInstances } = await client.send(command);
    const instanceList = TerminatingInstances.map(
      (instance) => ` • ${instance.InstanceId}`,
    );
    console.log("Terminating instances:");
    console.log(instanceList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- APIの詳細については、「AWS SDK for JavaScript API リファレンス」の「[TerminateInstances](#)」を参照してください。

Kotlin

SDK for Kotlin

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[TerminateInstances](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたインスタスを終了します (インスタスが実行中または「停止状態」である可能性があります)。コマンドレットは、続行する前に確認を求めるプロンプトを表示します。プロンプトを表示しないようにするには、-Force スイッチを使用します。

```
Remove-EC2Instance -InstanceId i-12345678
```

出力:

CurrentState	InstanceId	PreviousState
-----	-----	-----
Amazon.EC2.Model.InstanceState	i-12345678	Amazon.EC2.Model.InstanceState

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[TerminateInstances](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたインスタンスを終了します (インスタンスが実行中または「停止状態」である可能性があります)。コマンドレットは、続行する前に確認を求めるプロンプトを表示します。プロンプトを表示しないようにするには、-Force スイッチを使用します。

```
Remove-EC2Instance -InstanceId i-12345678
```

出力:

CurrentState	InstanceId	PreviousState
-----	-----	-----
Amazon.EC2.Model.InstanceState	i-12345678	Amazon.EC2.Model.InstanceState

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[TerminateInstances](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
    using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
```

```

) -> None:
    """
    Initializes the EC2InstanceWrapper with an EC2 client and optional
    instances.

    :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                        access to AWS EC2 services.
    :param instances: A list of dictionaries representing Boto3 Instance
objects. These are high-level objects that
                        wrap instance actions.
    """
    self.ec2_client = ec2_client
    self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def terminate(self) -> None:
        """
        Terminates instances and waits for them to reach the terminated state.
        """
        if not self.instances:
            logger.info("No instances to terminate.")
            return

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            self.ec2_client.terminate_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_terminated")
            waiter.wait(InstanceIds=instance_ids)
            self.instances.clear()
            for instance_id in instance_ids:
                print(f"• Instance ID: {instance_id}\n" f"• Action: Terminated")

```

```
except ClientError as err:
    logger.error(
        f"Failed instance termination details:\n\t{str(self.instances)}"
    )
    error_code = err.response["Error"]["Code"]
    if error_code == "InvalidInstanceID.NotFound":
        logger.error(
            "One or more instance IDs do not exist. "
            "Please verify the instance IDs and try again."
        )
    raise
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[TerminateInstances](#)」を参照してください。

Ruby

SDK for Ruby

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
require 'aws-sdk-ec2'

# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was terminated; otherwise, false.
# @example
#   exit 1 unless instance_terminated?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'i-123abc'
```

```
# )
def instance_terminated?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

  if response.instance_statuses.count.positive? &&
    response.instance_statuses[0].instance_state.name == 'terminated'

    puts 'The instance is already terminated.'
    return true
  end

  ec2_client.terminate_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_terminated, instance_ids: [instance_id])
  puts 'Instance terminated.'
  true
rescue StandardError => e
  puts "Error terminating instance: #{e.message}"
  false
end

# Example usage:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:  ruby ec2-ruby-example-terminate-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION '
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-terminate-instance-i-123abc.rb ' \
      'i-123abc us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)
```

```
puts "Attempting to terminate instance '#{instance_id}' " \
      '(this might take a few minutes)...'\
return if instance_terminated?(ec2_client, instance_id)

puts 'Could not terminate instance.'
end

run_me if $PROGRAM_NAME == __FILE__
```

- API の詳細については、AWS SDK for Ruby API リファレンスの「[TerminateInstances](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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(())
}
```

Writers API を使用して、インスタンスが終了状態になるまで待ちます。Writers API を使用するには、rust ファイルで `use aws\_sdk\_ec2::client::Writers` を使用する必要があります。

```

    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(())
    }

```

- APIの詳細については、「AWS SDK for Rust API リファレンス」の「[TerminateInstances](#)」を参照してください。

Swift

SDK for Swift

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```

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.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
    }
}
```

- APIの詳細については、「AWS SDK for Swift API リファレンス」の「[TerminateInstances](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で `UnassignPrivateIpAddresses` を使用する

次のサンプルコードは、`UnassignPrivateIpAddresses` を使用方法を説明しています。

CLI

AWS CLI

ネットワークインターフェイスからセカンダリプライベート IP アドレスの割り当てを解除するには

この例では、指定されたネットワークインターフェイスから指定されたプライベート IP アドレスの割り当てを解除します。コマンドが成功した場合、出力は返りません。

コマンド:

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

- API の詳細については、「AWS CLI コマンドリファレンス」の「[UnassignPrivateIpAddresses](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたネットワークインターフェイスから指定されたプライベート IP アドレスの割り当てを解除します。

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[UnassignPrivateIpAddresses](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたネットワークインターフェイスから指定されたプライベート IP アドレスの割り当てを解除します。

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

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[UnassignPrivateIpAddresses](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **UnmonitorInstances**で を使用する

次のサンプルコードは、UnmonitorInstances を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

C++

SDK for C++

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
//! Disable monitoring for an EC2 instance.  
/*!  
    \param instanceId: An EC2 instance ID.  
    \param clientConfiguration: AWS client configuration.  
    \return bool: Function succeeded.  
*/
```

```

bool AwsDoc::EC2::disableMonitoring(const Aws::String &instanceId,
                                     const Aws::Client::ClientConfiguration
                                     &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::UnmonitorInstancesRequest unrequest;
    unrequest.AddInstanceIds(instanceId);
    unrequest.SetDryRun(true);

    Aws::EC2::Model::UnmonitorInstancesOutcome dryRunOutcome =
    ec2Client.UnmonitorInstances(unrequest);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
            << "Failed dry run to disable monitoring on instance. A dry run
            should trigger an error."
            <<
            std::endl;
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType() !=
        Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to disable monitoring on instance " <<
            instanceId << ": " << dryRunOutcome.GetError().GetMessage() <<
            std::endl;
        return false;
    }

    unrequest.SetDryRun(false);
    Aws::EC2::Model::UnmonitorInstancesOutcome unmonitorInstancesOutcome =
    ec2Client.UnmonitorInstances(unrequest);
    if (!unmonitorInstancesOutcome.IsSuccess()) {
        std::cout << "Failed to disable monitoring on instance " << instanceId
            << ": " << unmonitorInstancesOutcome.GetError().GetMessage() <<
            std::endl;
    } else {
        std::cout << "Successfully disable monitoring on instance " <<
            instanceId << std::endl;
    }

    return unmonitorInstancesOutcome.IsSuccess();
}

```

- APIの詳細については、AWS SDK for C++ API リファレンスの「[UnmonitorInstances](#)」を参照してください。

CLI

AWS CLI

インスタンスの詳細モニタリングを無効にするには

このコマンド例は、指定されたインスタンスの詳細モニタリングを無効にします。

コマンド:

```
aws ec2 unmonitor-instances --instance-ids i-1234567890abcdef0
```

出力:

```
{
  "InstanceMonitorings": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "Monitoring": {
        "State": "disabling"
      }
    }
  ]
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「[UnmonitorInstances](#)」を参照してください。

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

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

```
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

/**
 * Turn off detailed monitoring for the selected instance.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new UnmonitorInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { InstanceMonitorings } = await client.send(command);
    const instanceMonitoringsList = InstanceMonitorings.map(
      (im) =>
        ` • Detailed monitoring state for ${im.InstanceId} is
${im.Monitoring.State}.`,
    );
    console.log("Monitoring status:");
    console.log(instanceMonitoringsList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- API の詳細については、AWS SDK for JavaScript API リファレンスの「[UnmonitorInstances](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定されたインスタスの詳細モニタリングを無効にします。

```
Stop-EC2InstanceMonitoring -InstanceId i-12345678
```

出力:

InstanceId	Monitoring
i-12345678	Amazon.EC2.Model.Monitoring

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[UnmonitorInstances](#)」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定されたインスタンスの詳細モニタリングを無効にします。

```
Stop-EC2InstanceMonitoring -InstanceId i-12345678
```

出力:

InstanceId	Monitoring
i-12345678	Amazon.EC2.Model.Monitoring

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[UnmonitorInstances](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **UpdateSecurityGroupRuleDescriptionsIngress** を使用する

次のサンプルコードは、UpdateSecurityGroupRuleDescriptionsIngress を使用方法を説明しています。

CLI

AWS CLI

例 1: CIDR ソースを使用してインバウンドセキュリティグループのルールの説明を更新するには

次の `update-security-group-rule-descriptions-ingress` の例では、指定したポートと IPv4 アドレス範囲のセキュリティグループルールの説明を更新します。説明 'SSH access from ABC office' は、ルールの既存の説明を置き換えます。

```
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"}]'
```

出力:

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

詳細については、「Amazon EC2 ユーザーガイド」の「[セキュリティグループのルール](#)」を参照してください。

例 2: プレフィックスリストソースを使用してインバウンドセキュリティグループのルールの説明を更新するには

次の `update-security-group-rule-descriptions-ingress` の例では、指定したポートとプレフィックスリストのセキュリティグループルールの説明を更新します。説明 'SSH access from ABC office' は、ルールの既存の説明を置き換えます。

```
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"}]'
```

出力:

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

詳細については、「Amazon EC2 ユーザーガイド」の「[セキュリティグループのルール](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[UpdateSecurityGroupRuleDescriptionsIngress](#)」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: 進入 (受信) セキュリティグループルールの説明を更新します。

```
$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
```

例 2: 既存の進入 (受信) セキュリティグループルールの説明を削除します (リクエストでパラメータを省略)。

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupId
"sgr-1234567890"
$ruleWithoutDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
    "SecurityGroupId" = $existingInboundRule.SecurityGroupId
}

Update-EC2SecurityGroupRuleIngressDescription -GroupId
$existingInboundRule.GroupId -SecurityGroupRuleDescription
$ruleWithoutDescription
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[UpdateSecurityGroupRuleDescriptionsIngress](#)」を参照してください。

Tools for PowerShell V5

例 1: 進入 (受信) セキュリティグループルールの説明を更新します。

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupRuleId
"sgr-1234567890"
$ruleWithUpdatedDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
    "SecurityGroupRuleId" = $existingInboundRule.SecurityGroupRuleId
    "Description" = "Updated rule description"
}

Update-EC2SecurityGroupRuleIngressDescription -GroupId
$existingInboundRule.GroupId -SecurityGroupRuleDescription
$ruleWithUpdatedDescription
```

例 2: 既存の進入 (受信) セキュリティグループルールの説明を削除します (リクエストでパラメータを省略)。

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupRuleId
"sgr-1234567890"
$ruleWithoutDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
    "SecurityGroupRuleId" = $existingInboundRule.SecurityGroupRuleId
}

Update-EC2SecurityGroupRuleIngressDescription -GroupId
$existingInboundRule.GroupId -SecurityGroupRuleDescription
$ruleWithoutDescription
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[UpdateSecurityGroupRuleDescriptionsIngress](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

SDK を使用した Amazon EC2 のシナリオ AWS SDKs

次のコード例は、AWS SDKs を使用して Amazon EC2 で一般的なシナリオを実装する方法を示しています。これらのシナリオは、Amazon EC2 内で複数の関数を呼び出すか、その他の AWS のサービスと組み合わせることで、特定のタスクを実行する方法を示しています。各シナリオには、完全なソースコードへのリンクが含まれており、そこからコードの設定方法と実行方法に関する手順を確認できます。

シナリオは、サービスアクションをコンテキストで理解するのに役立つ中級レベルの経験を対象としています。

例

- [AWS SDK を使用して回復力のあるサービスを構築および管理します。](#)
- [CLI を使用してプライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成します。](#)
- [CLI を使用した Amazon VPC の開始方法](#)
- [CLI を使用した Transit Gateway の開始方法](#)
- [CLI を使用した Amazon VPC IPAM の開始方法](#)

AWS SDK を使用して回復力のあるサービスを構築および管理します。

次のコード例は、本、映画、曲のレコメンデーションを返すロードバランシングウェブサービスの作成方法を示しています。この例では、障害に対するサービスの対応方法と、障害発生時の耐障害性を高めるためにサービスを再構築する方法を示しています。

- Amazon EC2 Auto Scaling グループを使用して、起動テンプレートに基づいて Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを作成し、インスタンス数を所定の範囲内に維持します。
- Elastic Load Balancing で HTTP リクエストを処理して配信します。
- Auto Scaling グループ内のインスタンスの状態を監視し、正常なインスタンスにのみリクエストを転送します。
- 各 EC2 インスタンスで Python ウェブサーバーを実行して HTTP リクエストを処理します。ウェブサーバーはレコメンデーションとヘルスチェックを返します。
- Amazon DynamoDB テーブルを使用してレコメンデーションサービスをシミュレートできます。
- AWS Systems Manager パラメータを更新して、リクエストとヘルスチェックに対するウェブサーバーの応答を制御します。

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブなシナリオを実行します。

```
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;
    }
```

Auto Scaling と Amazon EC2 のアクションをラップするクラスを作成します。

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

Elastic Load Balancing のアクションをラップするクラスを作成します。

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

```

DynamoDB を使用してレコメンデーションサービスをシミュレートするクラスを作成します。

```

/// <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.Write(".");
        }
        while (status != "ACTIVE");

        return status == TableStatus.ACTIVE;
    }
    catch (ResourceInUseException)
    {
        Console.WriteLine($"Table {tableName} already exists.");
        return false;
    }
}

/// <summary>
/// Populate the database table with data from a specified path.
/// </summary>
/// <param name="databaseTableName">The name of the table.</param>
/// <param name="recommendationsPath">The path of the recommendations data.</
param>
/// <returns>Async task.</returns>
public async Task PopulateDatabase(string databaseTableName, string
recommendationsPath)
{
    var recommendationsText = await
File.ReadAllTextAsync(recommendationsPath);
    var records =

```

```

JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);
    var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

    foreach (var record in records!)
    {
        batchWrite.AddPutItem(record);
    }

    await batchWrite.ExecuteAsync();
}

/// <summary>
/// Delete the recommendation table by name.
/// </summary>
/// <param name="tableName">The name of the recommendation table.</param>
/// <returns>Async task.</returns>
public async Task DestroyDatabaseByName(string tableName)
{
    try
    {
        await _amazonDynamoDb.DeleteTableAsync(
            new DeleteTableRequest() { TableName = tableName });
        Console.WriteLine($"Table {tableName} was deleted.");
    }
    catch (ResourceNotFoundException)
    {
        Console.WriteLine($"Table {tableName} not found");
    }
}
}

```

Systems Manager のアクションをラップするクラスを作成します。

```

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

- APIの詳細については、「AWS SDK for .NET API リファレンス」の以下のトピックを参照してください。

- [AttachLoadBalancerTargetGroups](#)
- [CreateAutoScalingGroup](#)
- [CreateInstanceProfile](#)
- [CreateLaunchTemplate](#)
- [CreateListener](#)
- [CreateLoadBalancer](#)
- [CreateTargetGroup](#)
- [DeleteAutoScalingGroup](#)
- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)

- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplacelamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブなシナリオを実行します。

```
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);
}
}

```

Auto Scaling と Amazon EC2 のアクションをラップするクラスを作成します。

```
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()
    .getInstanceProfileName(profileName)
    .roleName(roleName)
    .build();

getIAMClient().removeRoleFromInstanceProfile(profileRequest);
DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
    .getInstanceProfileName(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.");
}
}

```

Elastic Load Balancing のアクションをラップするクラスを作成します。

```
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 "";
    }
}
```

DynamoDB を使用してレコメンデーションサービスをシミュレートするクラスを作成します。

```
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);
}
}
```

Systems Manager のアクションをラップするクラスを作成します。

```
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);
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。

- [AttachLoadBalancerTargetGroups](#)
- [CreateAutoScalingGroup](#)
- [CreateInstanceProfile](#)
- [CreateLaunchTemplate](#)
- [CreateListener](#)
- [CreateLoadBalancer](#)
- [CreateTargetGroup](#)
- [DeleteAutoScalingGroup](#)
- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplaceIamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)

- [UpdateAutoScalingGroup](#)

JavaScript

SDK for JavaScript (v3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブなシナリオを実行します。

```
#!/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.",
  });
}

```

すべてのリソースをデプロイするための手順を作成します。

```

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,
];

```

デモを実行するための手順を作成します。

```

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;
}
```

すべてのリソースを破棄するための手順を作成します。

```
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 (error) {
            console.error(error);
        }
    })
    );
}

```

```

    }},
  );
} catch (e) {
  state.removeRoleFromInstanceProfileError = e;
}
}},
new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
  if (state.removeRoleFromInstanceProfile) {
    console.error(state.removeRoleFromInstanceProfileError);
    return MESSAGES.removeRoleFromInstanceProfileError
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }
  return MESSAGES.removedRoleFromInstanceProfile
    .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
    .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
}),
new ScenarioAction("deleteInstanceRole", async (state) => {
  try {
    const client = new IAMClient({});
    await client.send(
      new DeleteRoleCommand({
        RoleName: NAMES.instanceRoleName,
      }),
    );
  } catch (e) {
    state.deleteInstanceRoleError = e;
  }
}),
new ScenarioOutput("deleteInstanceRoleResult", (state) => {
  if (state.deleteInstanceRoleError) {
    console.error(state.deleteInstanceRoleError);
    return MESSAGES.deleteInstanceRoleError.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    );
  }
  return MESSAGES.deletedInstanceRole.replace(
    "${INSTANCE_ROLE_NAME}",
    NAMES.instanceRoleName,
  );
}),
new ScenarioAction("deleteInstanceProfile", async (state) => {
  try {

```

```
const client = new IAMClient({});
await client.send(
  new DeleteInstanceProfileCommand({
    InstanceProfileName: NAMES.instanceProfileName,
  }),
);
} catch (e) {
  state.deleteInstanceProfileError = e;
}
}),
new ScenarioOutput("deleteInstanceProfileResult", (state) => {
  if (state.deleteInstanceProfileError) {
    console.error(state.deleteInstanceProfileError);
    return MESSAGES.deleteInstanceProfileError.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    );
  }
  return MESSAGES.deletedInstanceProfile.replace(
    "${INSTANCE_PROFILE_NAME}",
    NAMES.instanceProfileName,
  );
}),
new ScenarioAction("deleteAutoScalingGroup", async (state) => {
  try {
    await terminateGroupInstances(NAMES.autoScalingGroupName);
    await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
      await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
    });
  } catch (e) {
    state.deleteAutoScalingGroupError = e;
  }
}),
new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {
  if (state.deleteAutoScalingGroupError) {
    console.error(state.deleteAutoScalingGroupError);
    return MESSAGES.deleteAutoScalingGroupError.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    );
  }
  return MESSAGES.deletedAutoScalingGroup.replace(
    "${AUTO_SCALING_GROUP_NAME}",
    NAMES.autoScalingGroupName,
  );
});
```

```

    );
  }},
  new ScenarioAction("deleteLaunchTemplate", async (state) => {
    const client = new EC2Client({});
    try {
      await client.send(
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
      );
    } catch (e) {
      state.deleteLaunchTemplateError = e;
    }
  }},
  new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
    if (state.deleteLaunchTemplateError) {
      console.error(state.deleteLaunchTemplateError);
      return MESSAGES.deleteLaunchTemplateError.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    }
    return MESSAGES.deletedLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    );
  }},
  new ScenarioAction("deleteLoadBalancer", async (state) => {
    try {
      const client = new ElasticLoadBalancingV2Client({});
      const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
      await client.send(
        new DeleteLoadBalancerCommand({
          LoadBalancerArn: loadBalancer.LoadBalancerArn,
        }),
      );
      await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
        const lb = await findLoadBalancer(NAMES.loadBalancerName);
        if (lb) {
          throw new Error("Load balancer still exists.");
        }
      });
    } catch (e) {
      state.deleteLoadBalancerError = e;
    }
  })
);

```

```

    }
  })),
  new ScenarioOutput("deleteLoadBalancerResult", (state) => {
    if (state.deleteLoadBalancerError) {
      console.error(state.deleteLoadBalancerError);
      return MESSAGES.deleteLoadBalancerError.replace(
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    }
    return MESSAGES.deletedLoadBalancer.replace(
      "${LB_NAME}",
      NAMES.loadBalancerName,
    );
  })),
  new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    try {
      const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
          Names: [NAMES.loadBalancerTargetGroupName],
        }),
      );

      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        client.send(
          new DeleteTargetGroupCommand({
            TargetGroupArn: TargetGroups[0].TargetGroupArn,
          }),
        );
    } catch (e) {
      state.deleteLoadBalancerTargetGroupError = e;
    }
  })),
  new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
    if (state.deleteLoadBalancerTargetGroupError) {
      console.error(state.deleteLoadBalancerTargetGroupError);
      return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      );
    }
    return MESSAGES.deletedLoadBalancerTargetGroup.replace(

```

```

        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
    );
}),
new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
    try {
        const client = new IAMClient({});
        await client.send(
            new RemoveRoleFromInstanceProfileCommand({
                InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
                RoleName: NAMES.ssmOnlyRoleName,
            }),
        );
    } catch (e) {
        state.detachSsmOnlyRoleFromProfileError = e;
    }
}),
new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
    if (state.detachSsmOnlyRoleFromProfileError) {
        console.error(state.detachSsmOnlyRoleFromProfileError);
        return MESSAGES.detachSsmOnlyRoleFromProfileError
            .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
            .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    }
    return MESSAGES.detachedSsmOnlyRoleFromProfile
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
}),
new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
    try {
        const iamClient = new IAMClient({});
        const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
        await iamClient.send(
            new DetachRolePolicyCommand({
                RoleName: NAMES.ssmOnlyRoleName,
                PolicyArn: ssmOnlyPolicy.Arn,
            }),
        );
    } catch (e) {
        state.detachSsmOnlyCustomRolePolicyError = e;
    }
}),
new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
    if (state.detachSsmOnlyCustomRolePolicyError) {

```

```

        console.error(state.detachSsmOnlyCustomRolePolicyError);
        return MESSAGES.detachSsmOnlyCustomRolePolicyError
            .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
            .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    }
    return MESSAGES.detachedSsmOnlyCustomRolePolicy
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    })),
    new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
        try {
            const iamClient = new IAMClient({});
            await iamClient.send(
                new DetachRolePolicyCommand({
                    RoleName: NAMES.ssmOnlyRoleName,
                    PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
                }),
            );
        } catch (e) {
            state.detachSsmOnlyAWSRolePolicyError = e;
        }
    })),
    new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
        if (state.detachSsmOnlyAWSRolePolicyError) {
            console.error(state.detachSsmOnlyAWSRolePolicyError);
            return MESSAGES.detachSsmOnlyAWSRolePolicyError
                .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
                .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
        }
        return MESSAGES.detachedSsmOnlyAWSRolePolicy
            .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
            .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
    })),
    new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {
        try {
            const iamClient = new IAMClient({});
            await iamClient.send(
                new DeleteInstanceProfileCommand({
                    InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
                }),
            );
        } catch (e) {
            state.deleteSsmOnlyInstanceProfileError = e;
        }
    })

```



```

    }},
    new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
      if (state.deleteSsmOnlyInstanceProfileError) {
        console.error(state.deleteSsmOnlyInstanceProfileError);
        return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
          "${INSTANCE_PROFILE_NAME}",
          NAMES.ssmOnlyInstanceProfileName,
        );
      }
      return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.ssmOnlyInstanceProfileName,
      );
    }),
    new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
      try {
        const iamClient = new IAMClient({});
        const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
        await iamClient.send(
          new DeletePolicyCommand({
            PolicyArn: ssmOnlyPolicy.Arn,
          }),
        );
      } catch (e) {
        state.deleteSsmOnlyPolicyError = e;
      }
    }),
    new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
      if (state.deleteSsmOnlyPolicyError) {
        console.error(state.deleteSsmOnlyPolicyError);
        return MESSAGES.deleteSsmOnlyPolicyError.replace(
          "${POLICY_NAME}",
          NAMES.ssmOnlyPolicyName,
        );
      }
      return MESSAGES.deletedSsmOnlyPolicy.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
      );
    }),
    new ScenarioAction("deleteSsmOnlyRole", async (state) => {
      try {
        const iamClient = new IAMClient({});
        await iamClient.send(

```

```

        new DeleteRoleCommand({
            RoleName: NAMES.ssmOnlyRoleName,
        }),
    );
} catch (e) {
    state.deleteSsmOnlyRoleError = e;
}
}),
new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
    if (state.deleteSsmOnlyRoleError) {
        console.error(state.deleteSsmOnlyRoleError);
        return MESSAGES.deleteSsmOnlyRoleError.replace(
            "${ROLE_NAME}",
            NAMES.ssmOnlyRoleName,
        );
    }
    return MESSAGES.deletedSsmOnlyRole.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
    );
}),
new ScenarioAction(
    "revokeSecurityGroupIngress",
    async (
        /** @type {{ myIp: string, defaultSecurityGroup: { GroupId: string } }} */
        state,
    ) => {
        const ec2Client = new EC2Client({});

        try {
            await ec2Client.send(
                new RevokeSecurityGroupIngressCommand({
                    GroupId: state.defaultSecurityGroup.GroupId,
                    CidrIp: `${state.myIp}/32`,
                    FromPort: 80,
                    ToPort: 80,
                    IpProtocol: "tcp",
                }),
            );
        } catch (e) {
            state.revokeSecurityGroupIngressError = e;
        }
    },
),

```

```

    new ScenarioOutput("revokeSecurityGroupIngressResult", (state) => {
      if (state.revokeSecurityGroupIngressError) {
        console.error(state.revokeSecurityGroupIngressError);
        return MESSAGES.revokeSecurityGroupIngressError.replace(
          "${IP}",
          state.myIp,
        );
      }
      return MESSAGES.revokedSecurityGroupIngress.replace("${IP}", state.myIp);
    }),
  ],
);

/**
 * @param {string} policyName
 */
async function findPolicy(policyName) {
  const client = new IAMClient({});
  const paginatedPolicies = paginateListPolicies({ client }, {});
  for await (const page of paginatedPolicies) {
    const policy = page.Policies.find((p) => p.PolicyName === policyName);
    if (policy) {
      return policy;
    }
  }
}

/**
 * @param {string} groupName
 */
async function deleteAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  try {
    await client.send(
      new DeleteAutoScalingGroupCommand({
        AutoScalingGroupName: groupName,
      }),
    );
  } catch (err) {
    if (!(err instanceof Error)) {
      throw err;
    }
    console.log(err.name);
    throw err;
  }
}

```

```

}

/**
 * @param {string} groupName
 */
async function terminateGroupInstances(groupName) {
  const autoScalingClient = new AutoScalingClient({});
  const group = await findAutoScalingGroup(groupName);
  await autoScalingClient.send(
    new UpdateAutoScalingGroupCommand({
      AutoScalingGroupName: group.AutoScalingGroupName,
      MinSize: 0,
    }),
  );
  for (const i of group.Instances) {
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: i.InstanceId,
          ShouldDecrementDesiredCapacity: true,
        }),
      ),
    );
  }
}

async function findAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
  for await (const page of paginatedGroups) {
    const group = page.AutoScalingGroups.find(
      (g) => g.AutoScalingGroupName === groupName,
    );
    if (group) {
      return group;
    }
  }
  throw new Error(`Auto scaling group ${groupName} not found.`);
}

```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の以下のトピックを参照してください。

- [AttachLoadBalancerTargetGroups](#)
- [CreateAutoScalingGroup](#)
- [CreateInstanceProfile](#)
- [CreateLaunchTemplate](#)
- [CreateListener](#)
- [CreateLoadBalancer](#)
- [CreateTargetGroup](#)
- [DeleteAutoScalingGroup](#)
- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplaceIamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブなシナリオを実行します。

```
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()

```

Auto Scaling と Amazon EC2 のアクションをラップするクラスを作成します。

```

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}")

```

Elastic Load Balancing のアクションをラップするクラスを作成します。

```

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"]

```

DynamoDB を使用してレコメンデーションサービスをシミュレートするクラスを作成します。

```

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}."
        )

```

Systems Manager のアクションをラップするクラスを作成します。

```

class ParameterHelper:
    """
    Encapsulates Systems Manager parameters. This example uses these parameters
    to drive
    the demonstration of resilient architecture, such as failure of a dependency
    or
    how the service responds to a health check.
    """

    table: str = "doc-example-resilient-architecture-table"
    failure_response: str = "doc-example-resilient-architecture-failure-response"
    health_check: str = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name: str, ssm_client: boto3.client):
        """
        Initializes the ParameterHelper class with the necessary parameters.

        :param table_name: The name of the DynamoDB table that is used as a
        recommendation
                           service.
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.table_name = table_name

    def reset(self) -> None:
        """
        Resets the Systems Manager parameters to starting values for the demo.
        These are the name of the DynamoDB recommendation table, no response when
        a
        dependency fails, and shallow health checks.

```

```

    """
    self.put(self.table, self.table_name)
    self.put(self.failure_response, "none")
    self.put(self.health_check, "shallow")

def put(self, name: str, value: str) -> None:
    """
    Sets the value of a named Systems Manager parameter.

    :param name: The name of the parameter.
    :param value: The new value of the parameter.
    :raises ParameterHelperError: If the parameter value cannot be set.
    """
    try:
        self.ssm_client.put_parameter(
            Name=name, Value=value, Overwrite=True, Type="String"
        )
        log.info("Setting parameter %s to '%s'.", name, value)
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(f"Failed to set parameter {name}.")
        if error_code == "ParameterLimitExceeded":
            log.error(
                "The parameter limit has been exceeded. "
                "Consider deleting unused parameters or request a limit
increase."
            )
        elif error_code == "ParameterAlreadyExists":
            log.error(
                "The parameter already exists and overwrite is set to False.
"
                "Use Overwrite=True to update the parameter."
            )
        log.error(f"Full error:\n\t{err}")

```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の以下のトピックを参照してください。
 - [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](#)

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI を使用してプライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成します。

次のコード例は、以下の操作方法を示しています。

- CLI を使用してプライベートサブネットおよび NAT ゲートウェイを持つ VPC を作成します。
- VPC、サブネット、ルートテーブル、NAT ゲートウェイなど、必要なコンポーネントを設定します。
- 適切なアクセスとセキュリティのためにセキュリティグループと IAM ロールを設定します。
- CLI コマンドを使用して、これらのリソースの作成と設定を自動化します。

Bash

AWS CLI Bash スクリプトを使用する

Note

GitHub には、その他のリソースもあります。[デベロッパーチュートリアル](#)のサンプルレポジトリで用例一覧を検索し、設定と実行の方法を確認してください。

```
#!/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..."
    fi
}
```

```
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:
\(.Target.Id), 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
```

- API の詳細については、AWS CLI コマンドリファレンスの以下のトピックを参照してください。
 - [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](#)

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI を使用した Amazon VPC の開始方法

次のコード例は、以下の操作方法を示しています。

- アカウントの設定
- VPC の作成と設定
- ネットワークを設定する
- セキュリティの設定
- リソースのデプロイ
- テストと検証
- リソースをクリーンアップする
- 本番稼働への影響を考慮する
- セキュリティへの影響を考慮する

Bash

AWS CLI Bash スクリプトを使用する

Note

GitHub には、その他のリソースもあります。[デベロッパーチュートリアル](#)のサンプルレポジトリで用例一覧を検索し、設定と実行の方法を確認してください。

```
#!/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)"
```

- API の詳細については、AWS CLI コマンドリファレンスの以下のトピックを参照してください。
 - [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](#)

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI を使用した Transit Gateway の開始方法

次のコード例は、以下の操作方法を示しています。

- DNS サポートとデフォルトのルートテーブル設定を使用して Transit Gateway を作成する
- Transit Gateway が使用可能になるまで待機する
- サブネットを使用して Transit Gateway に 2 つの VPC をアタッチする
- VPC アタッチメントが使用可能になるまで待機する
- Transit Gateway を介して VPC の間にルートを追加する
- VPC リソース間の接続をテストする
- ルート、アタッチメント、Transit Gateway などのリソースをクリーンアップする

Bash

AWS CLI Bash スクリプトを使用する

Note

GitHub には、その他のリソースもあります。[デベロッパーチュートリアル](#)のサンプルレポジトリで用例一覧を検索し、設定と実行の方法を確認してください。

```
#!/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=en
\
    --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."
```

- APIの詳細については、AWS CLI コマンドリファレンスの以下のトピックを参照してください。

- [CreateRoute](#)
- [CreateSubnet](#)
- [CreateTransitGateway](#)
- [CreateTransitGatewayVpcAttachment](#)
- [CreateVpc](#)
- [DeleteRoute](#)
- [DeleteTransitGateway](#)
- [DeleteTransitGatewayVpcAttachment](#)
- [DescribeAvailabilityZones](#)
- [DescribeRouteTables](#)
- [DescribeSubnets](#)
- [DescribeTransitGatewayAttachments](#)
- [DescribeTransitGatewayVpcAttachments](#)
- [DescribeTransitGateways](#)

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI を使用した Amazon VPC IPAM の開始方法

次のコード例は、以下の操作方法を示しています。

- CLI を使用して Amazon VPC IP Address Manager (IPAM) をセットアップおよび設定します。
- オペレーションリージョン (例: us-east-1、us-west-2) がある IPAM を作成します。
- IPAM のプライベートスコープ ID を取得します。
- IPv4 プール (最上位、リージョン、開発プール) の階層構造を作成します。
- CIDR ブロックを各プールにプロビジョニングします (例: 10.0.0.0/8、10.0.0.0/16、10.0.0.0/24)。
- IPAM プールから割り当てられた CIDR を使用して VPC を作成します。
- IPAM プールの割り当てと VPC の作成を確認します。
- アクセス許可エラー、CIDR 割り当ての失敗、依存関係違反などの一般的な問題をトラブルシューティングします。
- 不要な料金が発生しないように、IPAM リソース (VPC、プール、CIDR、IPAM) をクリーンアップします。
- 高度な IPAM 機能の次のステップについて説明します。

Bash

AWS CLI Bash スクリプトを使用する

Note

GitHub には、その他のリソースもあります。[デベロッパーチュートリアル](#)の[サンプル](#)レポジトリで用例一覧を検索し、設定と実行の方法を確認してください。

```
#!/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

    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
    done
}
```

```

        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
```

- APIの詳細については、AWS CLI コマンドリファレンスの以下のトピックを参照してください。
 - [CreateIpam](#)
 - [CreateIpamPool](#)
 - [CreateVpc](#)
 - [DeleteIpam](#)
 - [DeleteIpamPool](#)
 - [DeleteVpc](#)
 - [DeprovisionIpamPoolCidr](#)
 - [DescribeIpamPools](#)
 - [DescribeIpams](#)
 - [DescribeVpcs](#)
 - [GetIpamPoolAllocations](#)
 - [GetIpamPoolCidrs](#)

- [ProvisionIpamPoolCidr](#)

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK を使用して Amazon EC2 リソースを作成する](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

Amazon CloudWatch を使用して Amazon EC2 API リクエストをモニタリングする

Amazon CloudWatch を使用して Amazon EC2 API リクエストをモニタリングすることで、生データを収集し、リアルタイムに近い読み取り可能なメトリクスに加工することができます。これらのメトリクスは、Amazon EC2 API オペレーションの使用状況と結果を経時的に追跡する簡単な方法を提供します。この情報により、ウェブアプリケーションのパフォーマンスをよりの確に把握し、さまざまな問題を特定して診断できます。また、特定のしきい値を監視するアラームを設定し、これらのしきい値に達したときに通知を送信したり特定のアクションを実行したりできます。

CloudWatch の詳細については、「[Amazon CloudWatch ユーザーガイド](#)」を参照してください。

Important

Amazon EC2 API メトリクスはオプトイン機能です。この機能へのアクセスをリクエストする必要があります。詳細については、「[the section called “Amazon EC2 API メトリクスを有効にする”](#)」を参照してください。

内容

- [Amazon EC2 API メトリクスを有効にする](#)
- [Amazon EC2 API メトリクスおよびディメンション](#)
- [メトリクスデータ保持](#)
- [ユーザーに代わって行われるリクエストのモニタリング](#)
- [料金](#)
- [Amazon CloudWatch との連携](#)

Amazon EC2 API メトリクスを有効にする

以下の手順に従って、AWS アカountのこの機能へのアクセスをリクエストします。

この機能へのアクセスをリクエストするには

1. [AWS サポート センター](#)を開きます。
2. [ケースを作成] を選択します。

3. [Account and billing] (アカウントおよび請求) を選択します。
4. [サービス] で [一般情報と開始方法] を選択します。
5. カテゴリ で、「Using AWS & Services」を選択します。
6. [次のステップ: 追加情報] を選択します。
7. [Subject (件名)] に **Request access to Amazon EC2 API metrics** と入力します。
8. [説明] に **Please grant my account access to Amazon EC2 API metrics.**
Related page: <https://docs.aws.amazon.com/AWSEC2/latest/APIReference/monitor.html> と入力します。アクセスが必要なリージョンも含めます。
9. [次のステップ: 今すぐ解決するか、お問い合わせください] を選択します。
10. [お問い合わせ] タブをクリックし、連絡の際に希望する言語と連絡方法を選択します。
11. [Submit] を選択してください。

Amazon EC2 API メトリクスおよびディメンション

メトリクス

Amazon EC2 API メトリクスは AWS/EC2/API 名前空間に含まれています。次の表に、Amazon EC2 API リクエストで使用できるメトリクスを示します。

メトリクス	説明
ClientErrors	クライアントエラーによって失敗した API リクエストの数。 これらのエラーは通常、リクエストで誤ったパラメータや無効なパラメータを指定したり、アクションやリソースを使用するアクセス許可を持たないユーザーに代わってアクションやリソースを使用するなど、クライアントが行ったアクションによって発生します。 単位: 数
RequestLimitExceeded	アカウントに対して Amazon EC2 API で許可されている最大リクエストレートを超えた回数。

メトリクス	説明
AwsTriggeredRequestLimitExceeded	Amazon EC2 API リクエストは、サービスのパフォーマンスを維持するためにスロットリングされます。リクエストが抑制されている場合は、<code>Client.RequestLimitExceeded</code> エラーが発生します。 単位: 数 オペレーションの問題により適用 AWS されたスロットリングによって発生した失敗した API リクエストの数。 このスロットリングが発生する<code>Client.RequestLimitExceeded</code> と、オペレーションの問題により AWS リクエストがスロットリングされたことを示すエラーが表示されます。このスロットリングは一時的なもので、運用上の問題が解決されると自動的に解決されます。エクスポネンシャルバックオフと再試行ロジックを実装して、これらの一時的なスロットルを適切に処理します。 単位: 数
ServerErrors	内部サーバーエラーによって失敗した API リクエストの数。 これらのエラーは通常、AWS サーバー側のエラー、例外、または障害によって発生します。 単位: 数
SuccessfulCalls	成功した API リクエストの数。 単位: 数

ディメンション

Amazon EC2 のメトリクスデータは、すべての EC2 API アクションでフィルタリングできます。ディメンションの詳細については、「[Amazon CloudWatch の概念](#)」を参照してください。

メトリクスデータ保持

Amazon EC2 API メトリクスは 1 分間隔で CloudWatch に送信されます。CloudWatch では、次のようにメトリクスデータを保持します。

- 期間が 60 秒 (1 分) のデータポイントは、15 日間使用できます。
- 期間が 300 秒 (5 分) のデータポイントは、63 日間使用できます。
- 期間が 3600 秒 (1 時間) のデータポイントは、455 日 (15 か月) 間使用できます。

ユーザーに代わって行われるリクエストのモニタリング

AWS サービスにリンクされたロールによって行われたリクエストなど、ユーザーに代わってサービスによって行われた API リクエストは、API スロットリング制限にはカウントされず、アカウントの Amazon CloudWatch にメトリクスを送信しません。これらのリクエストは CloudWatch を使用してモニタリングできません。

サードパーティーサービスプロバイダーがユーザーに代わって行う API リクエストは、API スロットリング上限にカウントされ、アカウントの Amazon CloudWatch にメトリクスを送信します。これらのリクエストは CloudWatch を使用してモニタリングできます。

料金

CloudWatch の一般的な価格設定と料金が適用されます。Amazon EC2 API メトリクスの使用に追加料金はかかりません。詳細については、「[Amazon CloudWatch 料金表](#)」をご覧ください。

Amazon CloudWatch との連携

目次

- [CloudWatch メトリクスの表示](#)
- [CloudWatch アラームの作成](#)

CloudWatch メトリクスの表示

次の手順を使用して、Amazon EC2 API メトリクスを表示します。

前提条件

アカウントの Amazon EC2 API メトリクスへのアクセスを有効にする必要があります。詳細については、「[the section called “Amazon EC2 API メトリクスを有効にする”](#)」を参照してください。

Amazon EC2 コンソールを使用して Amazon EC2 API メトリクスを表示するには

1. CloudWatch コンソールの <https://console.aws.amazon.com/cloudwatch/> を開いてください。
2. ナビゲーションペインで、[メトリクス]、[すべてのメトリクス] を選択します。
3. [参照] タブの [メトリクス] で、メトリクスを表示する目的のリージョンを選択します。
4. EC2 のメトリクスの名前空間を選択してください。
5. メトリクスを表示するには、[インスタンス別メトリクス] などのメトリクスディメンションを選択します。

Note

メトリクスは、非アクティブ状態が 2 週間続くと非表示になります。過去 2 週間に新しいデータポイントを受信しなかった場合、コンソールに表示されなくなり、コンソール検索ボックスにメトリクス名またはディメンション名を入力すると表示されず、[list-metrics](#) AWS CLI コマンドによって返されません。これらのメトリクスを取得するには、[get-metric-data](#) または [get-metric-statistics](#) コマンドを使用します。

コマンドラインを使用して Amazon EC2 API メトリクスを表示するには

以下のいずれかのコマンドを使用します。

- [list-metrics](#) (AWS CLI)

```
aws cloudwatch list-metrics --namespace "AWS/EC2/API"
```

- [Get-CWMetricList](#) (AWS Tools for Windows PowerShell)

```
Get-CWMetricList -Namespace "AWS/EC2/API"
```

CloudWatch アラームの作成

アラームの状態が変わったら、Amazon SNS メッセージを送信する Amazon CloudWatch のアラームを作成することができます。1 つのアラームで、指定した期間中、1 つのメトリクスを監視します。このアラームは、複数の期間にわたる一定のしきい値とメトリクスの値の関係性に基づき、SNS トピックに通知を送信します。

例えば、サーバー側のエラーが原因で失敗する DescribeInstances API リクエストの数をモニタリングするアラームを作成できます。次のアラームは、DescribeInstances API リクエストの失敗数が 5 分間に 10 個のサーバー側のエラーのしきい値に達したときに E メール通知を送信します。

前提条件

アカウントの Amazon EC2 API メトリクスへのアクセスを有効にする必要があります。詳細については、「[the section called “Amazon EC2 API メトリクスを有効にする”](#)」を参照してください。

Amazon EC2 DescribeInstances API リクエストサーバーエラーのアラームを作成するには

1. CloudWatch コンソールの <https://console.aws.amazon.com/cloudwatch/> を開いてください。
2. ナビゲーションペインで、[アラーム]、[すべてのアラーム] の順に選択します。
3. [アラームの作成] を選択します。
4. [メトリクスの選択] を選択し、以下の操作を行います。
 - a. [EC2/API] を選択します。
 - b. [アクションあたりのメトリクス] を選択します。
 - c. ServerErrors メトリクス名と同じ行にある DescribeInstances の横にあるチェックボックスをオンにします。
 - d. [メトリクスの選択] を選択します。
5. [Specify metric and conditions (メトリクスと条件の指定)] ページに、選択したメトリクスと統計のグラフや他の情報が表示されます。
 - a. [メトリクス] で、以下を指定します。
 - i. [統計] で、[合計] を選択します。
 - ii. [期間] で、[5 分] が選択されていることを確認します。
 - b. [Conditions (条件)] で、次のように指定します。
 - i. [Threshold type (しきい値タイプ)] で [静的] を選択します。

- ii. [ServerErrors が次の場合] で [以上] ($\geq$) を選択します。
 - iii. [...より大きい] に 10 と入力します。
 - c. [次へ] を選択します。
6. [Configure actions] (アクションの設定) ページが表示されます。
 - [通知] で、以下を指定します。
 - i. [アラーム状態トリガー] で、[アラーム状態] を選択します。
 - ii. [SNS トピックの選択] で、[既存の SNS トピックを選択] または [新しいトピックを作成] を選択し、通知の必須フィールドに入力します。
 - iii. [次へ] を選択します。
7. [名前と説明を追加] ページが表示されます。
 - a. [アラーム名] に、アラームの名前を入力します。名前には ASCII 文字のみを使用します。
 - b. [アラームの説明] に、アラームの説明を入力します (省略可能)。
 - c. [次へ] を選択します。
8. [プレビューと作成] ページが表示されます。ルーティング情報が正しいことを確認し、[アラームを作成] を選択します。

詳細については、『Amazon CloudWatch ユーザーガイド』の「[Amazon CloudWatch アラームの使用](#)」を参照してください。

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。