



Guide du développeur

Amazon Elastic Compute Cloud



Amazon Elastic Compute Cloud: Guide du développeur

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

Les marques commerciales et la présentation commerciale d'Amazon ne peuvent pas être utilisées en relation avec un produit ou un service extérieur à Amazon, d'une manière susceptible d'entraîner une confusion chez les clients, ou d'une manière qui dénigre ou discrédite Amazon. Toutes les autres marques commerciales qui ne sont pas la propriété d'Amazon appartiennent à leurs propriétaires respectifs, qui peuvent ou non être affiliés ou connectés à Amazon, ou sponsorisés par Amazon.

Table of Contents

Accès programmatique à Amazon EC2	1
Points de terminaison de service	1
Points de terminaison IPv4	7
Points de terminaison à double pile (IPv4 et IPv6)	7
Spécification des points de terminaison	8
Cohérence à terme	10
Idempotence	12
L'impuissance dans Amazon EC2	12
RunInstances idempuissance	16
Exemples	17
Recommandations de nouvelle tentative pour les demandes idempotentes	19
Limitation des demandes d'API	20
Comment l'étranglement est appliqué	20
Limites d'étranglement	22
Surveiller la régulation des API	29
Réessais et recul exponentiel	30
Demander une augmentation de limite	31
À l'aide du AWS CLI	32
En savoir plus sur le AWS CLI	32
En utilisant AWS CloudFormation	33
Amazon EC2 et modèles AWS CloudFormation	33
Ressources pour Amazon EC2	33
En savoir plus sur AWS CloudFormation	37
Utilisation d'un AWS SDK	38
Exemples de code pour l'API Amazon EC2	38
En savoir plus sur les AWS SDK	38
API de bas niveau pour Amazon EC2	39
Console-to-Code	40
Comment ça marche	40
Limites	41
Régions prises en charge	41
Formats de code pris en charge	41
Actions conservées	41
Tableau Actions enregistrées	42

Utilisation de Console-to-Code	42
Exemples de code	46
Actions	62
AcceptVpcPeeringConnection	68
AllocateAddress	70
AllocateHosts	82
AssignPrivateIpAddresses	85
AssociateAddress	86
AssociateDhcpOptions	100
AssociateRouteTable	101
AttachInternetGateway	102
AttachNetworkInterface	103
AttachVolume	105
AttachVpnGateway	106
AuthorizeSecurityGroupEgress	107
AuthorizeSecurityGroupIngress	110
CancelCapacityReservation	130
CancelImportTask	132
CancelSpotFleetRequests	133
CancelSpotInstanceRequests	135
ConfirmProductInstance	136
CopyImage	137
CopySnapshot	139
CreateCapacityReservation	141
CreateCustomerGateway	145
CreateDhcpOptions	146
CreateFlowLogs	149
CreateImage	151
CreateInstanceExportTask	154
CreateInternetGateway	156
CreateKeyPair	157
CreateLaunchTemplate	173
CreateNetworkAcl	182
CreateNetworkAclEntry	184
CreateNetworkInterface	185
CreatePlacementGroup	191

CreateRoute	192
CreateRouteTable	193
CreateSecurityGroup	198
CreateSnapshot	219
CreateSpotDatafeedSubscription	222
CreateSubnet	223
CreateTags	230
CreateVolume	234
CreateVpc	238
CreateVpcEndpoint	245
CreateVpnConnection	249
CreateVpnConnectionRoute	255
CreateVpnGateway	256
DeleteCustomerGateway	258
DeleteDhcpOptions	259
DeleteFlowLogs	260
DeleteInternetGateway	261
DeleteKeyPair	262
DeleteLaunchTemplate	273
DeleteNetworkAcl	277
DeleteNetworkAclEntry	278
DeleteNetworkInterface	279
DeletePlacementGroup	281
DeleteRoute	282
DeleteRouteTable	283
DeleteSecurityGroup	284
DeleteSnapshot	294
DeleteSpotDatafeedSubscription	296
DeleteSubnet	297
DeleteTags	298
DeleteVolume	300
DeleteVpc	301
DeleteVpnConnection	302
DeleteVpnConnectionRoute	303
DeleteVpnGateway	305
DeregisterImage	306

DescribeAccountAttributes	307
DescribeAddresses	310
DescribeAvailabilityZones	319
DescribeBundleTasks	326
DescribeCapacityReservations	328
DescribeCustomerGateways	331
DescribeDhcpOptions	333
DescribeFlowLogs	337
DescribeHostReservationOfferings	339
DescribeHosts	342
DescribeIamInstanceProfileAssociations	344
DescribeIdFormat	349
DescribeIdentityIdFormat	350
DescribeImageAttribute	352
DescribeImages	355
DescribeImportImageTasks	365
DescribeImportSnapshotTasks	368
DescribeInstanceAttribute	371
DescribeInstanceState	375
DescribeInstanceTypes	378
DescribeInstances	392
DescribeInternetGateways	421
DescribeKeyPairs	423
DescribeNetworkAcls	433
DescribeNetworkInterfaceAttribute	438
DescribeNetworkInterfaces	441
DescribePlacementGroups	446
DescribePrefixLists	448
DescribeRegions	449
DescribeRouteTables	463
DescribeScheduledInstanceAvailability	467
DescribeScheduledInstances	469
DescribeSecurityGroups	471
DescribeSnapshotAttribute	487
DescribeSnapshots	489
DescribeSpotDatafeedSubscription	495

DescribeSpotFleetInstances	496
DescribeSpotFleetRequestHistory	497
DescribeSpotFleetRequests	500
DescribeSpotInstanceRequests	504
DescribeSpotPriceHistory	508
DescribeSubnets	511
DescribeTags	519
DescribeVolumeAttribute	525
DescribeVolumeStatus	526
DescribeVolumes	528
DescribeVpcAttribute	532
DescribeVpcClassicLink	535
DescribeVpcClassicLinkDnsSupport	536
DescribeVpcEndpointServices	538
DescribeVpcEndpoints	542
DescribeVpcs	546
DescribeVpnConnections	553
DescribeVpnGateways	556
DetachInternetGateway	558
DetachNetworkInterface	559
DetachVolume	560
DetachVpnGateway	562
DisableVgwRoutePropagation	563
DisableVpcClassicLink	564
DisableVpcClassicLinkDnsSupport	565
DisassociateAddress	566
DisassociateRouteTable	574
EnableVgwRoutePropagation	575
EnableVolumeIo	576
EnableVpcClassicLink	577
EnableVpcClassicLinkDnsSupport	578
GetConsoleOutput	579
GetHostReservationPurchasePreview	581
GetPasswordData	583
ImportImage	585
ImportKeyPair	587

ImportSnapshot	589
ModifyCapacityReservation	591
ModifyHosts	592
ModifyIdFormat	594
ModifyImageAttribute	596
ModifyInstanceAttribute	598
ModifyInstanceCreditSpecification	601
ModifyNetworkInterfaceAttribute	603
ModifyReservedInstances	605
ModifySnapshotAttribute	607
ModifySpotFleetRequest	608
ModifySubnetAttribute	610
ModifyVolumeAttribute	611
ModifyVpcAttribute	612
MonitorInstances	614
MoveAddressToVpc	618
PurchaseHostReservation	620
PurchaseScheduledInstances	621
RebootInstances	624
RegisterImage	634
RejectVpcPeeringConnection	636
ReleaseAddress	637
ReleaseHosts	648
ReplaceIamInstanceProfileAssociation	649
ReplaceNetworkAclAssociation	656
ReplaceNetworkAclEntry	657
ReplaceRoute	658
ReplaceRouteTableAssociation	659
ReportInstanceStatus	660
RequestSpotFleet	661
RequestSpotInstances	666
ResetImageAttribute	671
ResetInstanceAttribute	672
ResetNetworkInterfaceAttribute	674
ResetSnapshotAttribute	675
RevokeSecurityGroupEgress	676

RevokeSecurityGroupIngress	678
RunInstances	680
RunScheduledInstances	702
StartInstances	705
StopInstances	721
TerminateInstances	737
UnassignPrivateIpAddresses	749
UnmonitorInstances	750
Scénarios	754
Créer et gérer un service résilient	754
Démarrer avec les instances	915
Surveillez les demandes d'API à l'aide CloudWatch	1054
Activer les métriques de l'API Amazon EC2	1054
Mesures et dimensions de l'API Amazon EC2	1055
Métriques	1055
Dimensions	1056
Conservation des données métriques	1057
Suivi des demandes effectuées en votre nom	1057
Facturation	1057
Travailler avec Amazon CloudWatch	1057
Afficher CloudWatch les métriques	1058
Création d' CloudWatch alarmes	1058
.....	mlxi

Accès programmatique à Amazon EC2

Vous pouvez créer et gérer vos ressources Amazon EC2 à l'aide de l'interface de programmation AWS Management Console ou d'une interface de programmation. Pour plus d'informations sur l'utilisation de la console Amazon EC2, consultez le guide de l'utilisateur [Amazon EC2](#).

Comment ça marche

- [Points de terminaison Amazon EC2](#)
- [Cohérence éventuelle](#)
- [Idempotentié](#)
- [Limitation des demandes](#)

Interfaces de programmation

- [AWS Command Line Interface \(AWS CLI\)](#)
- [AWS CloudFormation](#)
- [AWS Kits SDK](#)
- [API de bas niveau](#)

Premiers pas

- [Exemples de code](#)
- [Console-to-Code](#)

Surveillance

- [AWS CloudTrail](#)
- [Surveiller les demandes](#)

Points de terminaison du service Amazon EC2

Un point de terminaison est une URL qui sert de point d'entrée à un service AWS Web. Amazon EC2 prend en charge les types de terminaux suivants :

- Points de terminaison IPv4
- Points de terminaison à double pile prenant en charge à la fois IPv4 et IPv6
- Points de terminaison FIPS

Lorsque vous faites une demande, vous pouvez spécifier le point de terminaison et la région à utiliser. Si vous ne spécifiez pas de point de terminaison, le point de terminaison IPv4 est utilisé par défaut. Pour utiliser un autre type de point de terminaison, vous devez le spécifier dans votre demande. Pour obtenir un exemple de la façon de procéder, consultez [Spécification des points de terminaison](#).

Nom de la région	Région	Point de terminaison	Protocole
US East (Ohio)	us-east-2	ec2.us-east-2.amazonaws.com	HTTP et HTTPS
		ec2-fips.us-east-2.amazonaws.com	HTTPS
		ec2.us-east-2.api.aws	HTTPS
US East (N. Virginia)	us-east-1	ec2.us-east-1.amazonaws.com	HTTP et HTTPS
		ec2-fips.us-east-1.amazonaws.com	HTTPS
		ec2.us-east-1.api.aws	HTTPS
USA Ouest (Californie du Nord)	us-west-1	ec2.us-west-1.amazonaws.com	HTTP et HTTPS
		ec2-fips.us-west-1.amazonaws.com	HTTPS
		ec2.us-west-1.api.aws	HTTPS
US West (Oregon)	us-west-2	ec2.us-west-2.amazonaws.com	HTTP et HTTPS
		ec2-fips.us-west-2.amazonaws.com	HTTPS
		ec2.us-west-2.api.aws	HTTPS

Nom de la région	Région	Point de terminaison	Protocole
			HTTPS
Afrique (Le Cap)	af-south-1	ec2.af-south-1.amazonaws.com ec2.af-south-1.api.aws	HTTP et HTTPS HTTPS
Asie-Pacifique (Hong Kong)	ap-east-1	ec2.ap-east-1.amazonaws.com ec2.ap-east-1.api.aws	HTTP et HTTPS HTTPS
Asie-Pacifique (Hyderabad)	ap-south-2	ec2.ap-south-2.amazonaws.com	HTTPS
Asie-Pacifique (Jakarta)	ap-southeast-3	ec2.ap-southeast-3.amazonaws.com	HTTPS
Asie-Pacifique (Melbourne)	ap-southeast-4	ec2.ap-southeast-4.amazonaws.com	HTTPS
Asia Pacific (Mumbai)	ap-south-1	ec2.ap-south-1.amazonaws.com ec2.ap-south-1.api.aws	HTTP et HTTPS HTTPS
Asie-Pacifique (Osaka)	ap-northeast-3	ec2.ap-northeast-3.amazonaws.com	HTTP et HTTPS

Nom de la région	Région	Point de terminaison	Protocole
Asia Pacific (Seoul)	ap-northeast-2	ec2.ap-northeast-2.amazonaws.com ec2.ap-northeast-2.api.aws	HTTP et HTTPS HTTPS
Asie-Pacifique (Singapour)	ap-southeast-1	ec2.ap-southeast-1.amazonaws.com ec2.ap-southeast-1.api.aws	HTTP et HTTPS HTTPS
Asia Pacific (Sydney)	ap-southeast-2	ec2.ap-southeast-2.amazonaws.com ec2.ap-southeast-2.api.aws	HTTP et HTTPS HTTPS
Asie-Pacifique (Tokyo)	ap-northeast-1	ec2.ap-northeast-1.amazonaws.com ec2.ap-northeast-1.api.aws	HTTP et HTTPS HTTPS
Canada (Central)	ca-central-1	ec2.ca-central-1.amazonaws.com ec2-fips.ca-central-1.amazonaws.com ec2.ca-central-1.api.aws	HTTP et HTTPS HTTPS HTTPS
Canada Ouest (Calgary)	ca-west-1	ec2.ca-west-1.amazonaws.com ec2-fips.ca-west-1.amazonaws.com	HTTPS HTTPS
Europe (Francfort)	eu-central-1	ec2.eu-central-1.amazonaws.com ec2.eu-central-1.api.aws	HTTP et HTTPS HTTPS

Nom de la région	Région	Point de terminaison	Protocole
Europe (Irlande)	eu-west-1	ec2.eu-west-1.amazonaws.com ec2.eu-west-1.api.aws	HTTP et HTTPS HTTPS
Europe (Londres)	eu-west-2	ec2.eu-west-2.amazonaws.com ec2.eu-west-2.api.aws	HTTP et HTTPS HTTPS
Europe (Milan)	eu-south-1	ec2.eu-south-1.amazonaws.com ec2.eu-south-1.api.aws	HTTP et HTTPS HTTPS
Europe (Paris)	eu-west-3	ec2.eu-west-3.amazonaws.com ec2.eu-west-3.api.aws	HTTP et HTTPS HTTPS
Europe (Espagne)	eu-south-2	ec2.eu-south-2.amazonaws.com	HTTPS
Europe (Stockholm)	eu-north-1	ec2.eu-north-1.amazonaws.com ec2.eu-north-1.api.aws	HTTP et HTTPS HTTPS
Europe (Zurich)	eu-central-2	ec2.eu-central-2.amazonaws.com	HTTPS
Israël (Tel Aviv)	il-central-1	ec2.il-central-1.amazonaws.com	HTTPS
Moyen-Orient (Bahreïn)	me-south-1	ec2.me-south-1.amazonaws.com ec2.me-south-1.api.aws	HTTP et HTTPS HTTPS

Nom de la région	Région	Point de terminaison	Protocole
Moyen-Orient (EAU)	me-central-1	ec2.me-central-1.amazonaws.com	HTTPS
Amérique du Sud (São Paulo)	sa-east-1	ec2.sa-east-1.amazonaws.com	HTTP et HTTPS
		ec2.sa-east-1.api.aws	HTTPS
AWS GovCloud (USA Est)	us-gov-east-1	ec2.us-gov-east-1.amazonaws.com	HTTPS
		ec2.us-gov-east-1.api.aws	HTTPS
AWS GovCloud (US-Ouest)	us-gov-west-1	ec2.us-gov-west-1.amazonaws.com	HTTPS
		ec2.us-gov-west-1.api.aws	HTTPS

Pour plus d'informations sur les régions, consultez [Régions et zones de disponibilité](#) dans le guide de l'utilisateur Amazon EC2. Pour obtenir la liste des points de terminaison pour Amazon EC2, [consultez Régions et points de terminaison](#) dans le manuel Amazon Web Services General Reference.

Rubriques

- [Points de terminaison IPv4](#)
- [Points de terminaison à double pile \(IPv4 et IPv6\)](#)
- [Spécification des points de terminaison](#)

Pour plus d'informations sur les points de terminaison FIPS, veuillez consulter [Points de terminaison FIPS](#) dans la Référence générale Amazon Web Services.

Points de terminaison IPv4

Les points de terminaison IPv4 prennent en charge le trafic IPv4 uniquement. Les points de terminaison IPv4 sont disponibles pour toutes les régions.

Si vous spécifiez le point de terminaison général, `ec2.amazonaws.com`, nous utilisons le point de terminaison pour `us-east-1`. Pour utiliser une autre région, spécifiez son point de terminaison associé. Par exemple, si vous le spécifiez `ec2.us-east-2.amazonaws.com` comme point de terminaison, nous dirigeons votre demande vers le point de terminaison `us-east-2`.

Les noms des points de terminaison IPv4 utilisent la convention d'affectation de noms suivante :

- `service.region.amazonaws.com`

Par exemple, le nom du point de terminaison IPv4 de la région `eu-west-1` est `ec2.eu-west-1.amazonaws.com`. Pour obtenir la liste des points de terminaison pour Amazon EC2, [consultez Régions et points de terminaison](#) dans le manuel Amazon Web Services General Reference.

Points de terminaison à double pile (IPv4 et IPv6)

Les points de terminaison à double pile prennent en charge à la fois le trafic IPv4 et le trafic IPv6. Les points de terminaison à double pile ne sont disponibles que dans les régions suivantes :

- `us-east-1`—Est des États-Unis (Virginie du Nord)
- `us-east-2`—Est des États-Unis (Ohio)
- `us-west-2`—Ouest des États-Unis (Oregon)
- `eu-west-1`—Europe (Irlande)
- `ap-south-1`—Asie-Pacifique (Mumbai)
- `sa-east-1`—Amérique du Sud (São Paulo)
- `us-gov-east-1`—AWS GovCloud (USA Est)
- `us-gov-west-1`—AWS GovCloud (US-Ouest)

Lorsque vous adressez une requête à un point de terminaison à double pile, l'URL du point de terminaison se résout en une adresse IPv6 ou IPv4, selon le protocole utilisé par votre réseau et votre client.

Amazon EC2 ne prend en charge que les points de terminaison régionaux à double pile, ce qui signifie que vous devez spécifier la région dans le nom du point de terminaison. Les noms des points de terminaison à double pile utilisent la convention d'affectation de noms suivante :

- `ec2.region.api.aws`

Par exemple, le nom du point de terminaison à double pile de la région eu-west-1 est `ec2.eu-west-1.api.aws`. Pour obtenir la liste des points de terminaison pour Amazon EC2, [consultez Régions et points de terminaison](#) dans le manuel Amazon Web Services General Reference.

Spécification des points de terminaison

Cette section fournit quelques exemples sur la manière de spécifier un point de terminaison lors de l'envoi d'une demande.

AWS CLI

Les exemples suivants montrent comment spécifier un point de terminaison pour la us-east-2 région à l'aide du AWS CLI.

- Double pile

```
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

Les exemples suivants montrent comment spécifier un point de terminaison pour la us-east-2 région à l'aide du AWS SDK for Java 2.x.

- Double pile

```
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

Les exemples suivants montrent comment spécifier un point de terminaison pour la eu-west-1 région à l'aide de la version AWS SDK for Java 1.x.

- Double pile

```
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

Les exemples suivants montrent comment spécifier un point de terminaison pour la us-east-1 région à l'aide du AWS SDK for Go.

- Double pile

```
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")
})
```

Cohérence éventuelle dans l'API Amazon EC2

L'API Amazon EC2 suit un modèle de cohérence final, en raison de la nature distribuée du système supportant l'API. Cela signifie que le résultat d'une commande d'API que vous exécutez et qui affecte vos ressources Amazon EC2 peut ne pas être immédiatement visible pour toutes les commandes suivantes que vous exécuterez. Vous devez garder cela à l'esprit lorsque vous exécutez une commande d'API qui suit immédiatement une commande d'API précédente.

La cohérence éventuelle peut affecter la façon dont vous gérez vos ressources. Par exemple, si vous exécutez une commande pour créer une ressource, elle sera éventuellement visible par les autres commandes. Cela signifie que si vous exécutez une commande pour modifier ou décrire la ressource que vous venez de créer, son identifiant ne s'est peut-être pas propagé dans tout le système et vous recevrez un message d'erreur indiquant que la ressource n'existe pas.

Pour gérer une éventuelle cohérence, vous pouvez effectuer les opérations suivantes :

- Vérifiez l'état de la ressource avant d'exécuter une commande pour la modifier. Exécutez la `Describe` commande appropriée à l'aide d'un algorithme de réduction exponentielle pour vous assurer de laisser suffisamment de temps à la commande précédente pour se propager dans le système. Pour ce faire, exécutez la `Describe` commande à plusieurs reprises, en commençant par un temps d'attente de quelques secondes, puis en augmentant progressivement jusqu'à cinq minutes.
- Ajoutez un temps d'attente entre les commandes suivantes, même si une `Describe` commande renvoie une réponse précise. Appliquez un algorithme de réduction exponentielle en commençant par quelques secondes d'attente, puis augmentez progressivement jusqu'à environ cinq minutes.

Exemples d'erreurs de cohérence éventuelles

Vous trouverez ci-dessous des exemples de codes d'erreur que vous pourriez rencontrer en raison d'une éventuelle cohérence.

- `InvalidInstanceID.NotFound`

Si vous exécutez la `RunInstances` commande avec succès, puis que vous exécutez immédiatement une autre commande en utilisant l'ID d'instance fourni dans la réponse de `RunInstances`, une `InvalidInstanceID.NotFound` erreur peut être renvoyée. Cela ne signifie pas que l'instance n'existe pas.

Certaines commandes spécifiques susceptibles d'être affectées sont les suivantes :

- `DescribeInstances`: pour confirmer l'état réel de l'instance, exécutez cette commande à l'aide d'un algorithme de rétrogradation exponentielle.
- `TerminateInstances`: Pour confirmer l'état de l'instance, exécutez d'abord la `DescribeInstances` commande à l'aide d'un algorithme de rétrogradation exponentielle.

 Important

Si un `InvalidInstanceID.NotFound` message d'erreur s'affiche après l'exécution `TerminateInstances`, cela ne signifie pas que l'instance est ou sera interrompue. Votre instance est peut-être toujours en cours d'exécution. C'est pourquoi il est important de confirmer d'abord l'état de l'instance à l'aide de `DescribeInstances`.

- `InvalidGroup.NotFound`

Si vous exécutez correctement la `CreateSecurityGroup` commande, puis que vous exécutez immédiatement une autre commande en utilisant l'ID de groupe de sécurité fourni dans la réponse de `CreateSecurityGroup`, une `InvalidGroup.NotFound` erreur peut être renvoyée. Pour confirmer l'état du groupe de sécurité, exécutez la `DescribeSecurityGroups` commande à l'aide d'un algorithme de réduction exponentielle.

- `InstanceLimitExceeded`

Vous avez demandé un nombre d'instances supérieur à ce que votre limite d'instances actuelle autorise pour le type d'instance spécifié. Vous pourriez atteindre cette limite de façon inattendue si vous lancez et mettez fin à des instances rapidement, car les instances résiliées sont prises en compte dans votre limite d'instances pendant un certain temps après leur résiliation.

Garantir l'idempotence des requêtes d'API Amazon EC2

Lorsque vous effectuez une demande d'API de mutation, la demande renvoie généralement un résultat avant la fin des flux de travail asynchrones de l'opération. Les opérations peuvent également expirer ou rencontrer d'autres problèmes de serveur avant d'être terminées, même si la demande a déjà renvoyé un résultat. Par conséquent, il peut être difficile de déterminer si la demande a abouti ou non et de multiples tentatives peuvent être déclenchées pour s'assurer que l'opération se termine correctement. Toutefois, si la demande initiale et les tentatives suivantes aboutissent, l'opération est terminée plusieurs fois. Cela signifie que vous pourriez créer plus de ressources que vous ne le souhaitez.

L'idempotence garantit qu'une demande d'API se termine correctement. Avec une demande idempotente, si la demande d'origine se termine avec succès, toutes les tentatives suivantes se terminent avec succès, sans aucune action supplémentaire. Cependant, le résultat peut contenir des informations mises à jour, telles que le statut de création actuel.

Table des matières

- [L'impuissance dans Amazon EC2](#)
- [RunInstances idempotence](#)
- [Exemples](#)
- [Recommandations de nouvelle tentative pour les demandes idempotentes](#)

L'impuissance dans Amazon EC2

Les actions d'API suivantes sont idempotentes par défaut et ne nécessitent aucune configuration supplémentaire. Les AWS CLI commandes correspondantes prennent également en charge l'idempotence par défaut.

Idempotent par défaut

- AssociateAddress
- CreateVpnConnection
- DisassociateAddress
- ReplaceNetworkAclAssociation
- TerminateInstances

Les actions d'API suivantes prennent éventuellement en charge l'idempotencie à l'aide d'un jeton client. Les AWS CLI commandes correspondantes prennent également en charge l'idempotencie à l'aide d'un jeton client. Un jeton client est une chaîne unique, distinguant majuscules et minuscules, comportant jusqu'à 64 caractères ASCII. Pour effectuer une demande d'API idempotente à l'aide de l'une de ces actions, spécifiez un jeton client dans la demande. Vous ne devez pas réutiliser le même jeton client pour d'autres demandes d'API. Si vous réessayez une demande qui s'est terminée avec succès en utilisant le même jeton client et les mêmes paramètres, la nouvelle tentative aboutit sans effectuer d'autre action. Si vous réessayez une demande réussie en utilisant le même jeton client, mais qu'un ou plusieurs paramètres sont différents, autres que la région ou la zone de disponibilité, la nouvelle tentative échoue avec une `IdempotentParameterMismatch` erreur.

Idempotent à l'aide d'un jeton client

- `AllocateHosts`
- `AllocateIpamPoolCidr`
- `AssociateClientVpnTargetNetwork`
- `AssociateIpamResourceDiscovery`
- `AttachVerifiedAccessTrustProvider`
- `AuthorizeClientVpnIngress`
- `CopyFpgaImage`
- `CopyImage`
- `CreateCapacityReservation`
- `CreateCapacityReservationFleet`
- `CreateClientVpnEndpoint`
- `CreateClientVpnRoute`
- `CreateEgressOnlyInternetGateway`
- `CreateFleet`
- `CreateFlowLogs`
- `CreateFpgaImage`
- `CreateInstanceConnectEndpoint`
- `CreateIpam`
- `CreateIpamPool`

- CreateIpamResourceDiscovery
- CreateIpamScope
- CreateLaunchTemplate
- CreateLaunchTemplateVersion
- CreateManagedPrefixList
- CreateNatGateway
- CreateNetworkAcl
- CreateNetworkInsightsAccessScope
- CreateNetworkInsightsPath
- CreateNetworkInterface
- CreateReplaceRootVolumeTask
- CreateReservedInstancesListing
- CreateRouteTable
- CreateTrafficMirrorFilter
- CreateTrafficMirrorFilterRule
- CreateTrafficMirrorSession
- CreateTrafficMirrorTarget
- CreateVerifiedAccessEndpoint
- CreateVerifiedAccessGroup
- CreateVerifiedAccessInstance
- CreateVerifiedAccessTrustProvider
- CreateVolume
- CreateVpcEndpoint
- CreateVpcEndpointConnectionNotification
- CreateVpcEndpointServiceConfiguration
- DeleteVerifiedAccessEndpoint
- DeleteVerifiedAccessGroup
- DeleteVerifiedAccessInstance

- DeleteVerifiedAccessTrustProvider
- DetachVerifiedAccessTrustProvider
- ExportImage
- ImportImage
- ImportSnapshot
- ModifyInstanceCreditSpecification
- ModifyLaunchTemplate
- ModifyReservedInstances
- ModifyVerifiedAccessEndpoint
- ModifyVerifiedAccessEndpointPolicy
- ModifyVerifiedAccessGroup
- ModifyVerifiedAccessGroupPolicy
- ModifyVerifiedAccessInstance
- ModifyVerifiedAccessInstanceLoggingConfiguration
- ModifyVerifiedAccessTrustProvider
- ProvisionIpamPoolCidr
- PurchaseHostReservation
- RequestSpotFleet
- RequestSpotInstances
- RunInstances
- StartNetworkInsightsAccessScopeAnalysis
- StartNetworkInsightsAnalysis

Types d'idempotence

- Régional — Les demandes sont idempotentes dans chaque région. Cependant, vous pouvez utiliser la même demande, y compris le même jeton client, dans une région différente.
- Zonal — Les demandes sont idempotentes dans chaque zone de disponibilité d'une région. Par exemple, si vous spécifiez le même jeton client lors de deux appels vers `AllocateHosts` la même région, les appels aboutissent s'ils spécifient des valeurs différentes pour le `AvailabilityZone` paramètre.

RunInstances idempuissance

L'action de [RunInstances](#) l'API utilise à la fois l'idempuissance régionale et zonale.

Le type d'idempuissance utilisé dépend de la manière dont vous spécifiez la zone de disponibilité dans votre demande d' RunInstances API. La demande utilise l'idempotencie zonale dans les cas suivants :

- Si vous spécifiez explicitement une zone de disponibilité à l'aide du AvailabilityZoneparamètre du type de données de placement
- Si vous spécifiez implicitement une zone de disponibilité à l'aide du paramètre SubnetId

Si vous ne spécifiez pas explicitement ou implicitement de zone de disponibilité, la demande utilise l'idempotencie régionale.

Idempotencie zonale

L'idempotencie zonale garantit qu'une demande d' RunInstances API est idempotente dans chaque zone de disponibilité d'une région. Cela garantit qu'une demande avec le même jeton client ne peut être traitée qu'une seule fois dans chaque zone de disponibilité d'une région. Toutefois, le même jeton client peut être utilisé pour lancer des instances dans d'autres zones de disponibilité de la région.

Par exemple, si vous envoyez une demande idempotente pour lancer une instance dans la zone de us-east-1a disponibilité, puis que vous utilisez le même jeton client dans une demande dans la zone de us-east-1b disponibilité, nous lançons des instances dans chacune de ces zones de disponibilité. Si un ou plusieurs paramètres sont différents, les tentatives suivantes avec le même jeton client dans ces zones de disponibilité sont soit renvoyées avec succès sans effectuer aucune autre action, soit échouer avec une IdempotentParameterMismatch erreur.

Idempuissance régionale

L'idempuissance régionale garantit qu'une demande d' RunInstances API est idempotente dans une région. Cela garantit qu'une demande avec le même jeton client ne peut être traitée qu'une seule fois dans une région. Cependant, la même demande, avec le même jeton client, peut être utilisée pour lancer des instances dans une autre région.

Par exemple, si vous envoyez une demande idempotente pour lancer une instance dans la us-east-1 région, puis que vous utilisez le même jeton client dans une demande dans la eu-

west-1 région, nous lançons des instances dans chacune de ces régions. Si un ou plusieurs paramètres sont différents, les tentatives suivantes avec le même jeton client dans ces régions sont soit renvoyées avec succès sans effectuer aucune autre action, soit échouer avec une `IdempotentParameterMismatch` erreur.

Tip

Si l'une des zones de disponibilité de la région demandée n'est pas disponible, les `RunInstances` demandes utilisant l'idempotence régionale peuvent échouer. Pour tirer parti des fonctionnalités de zone de disponibilité proposées par l'AWS infrastructure, nous vous recommandons d'utiliser l'idempotence zonale lors du lancement des instances. `RunInstances` les demandes qui utilisent l'idempotence zonale et ciblent une zone de disponibilité disponible aboutissent même si aucune autre zone de disponibilité dans la région demandée n'est disponible.

Exemples

AWS CLI exemples de commandes

Pour rendre une AWS CLI commande idempotente, ajoutez l'`--client-token` option.

Exemple 1 : Idempotence

La commande [allocate-hosts](#) suivante utilise l'idempotence car elle inclut un jeton client.

```
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
```

Exemple 2 : idempotence régionale des instances d'exécution

La commande [run-instances](#) suivante utilise l'idempotence régionale car elle inclut un jeton client mais ne spécifie pas explicitement ou implicitement de zone de disponibilité.

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

Exemple 3 : idempotence zonale des instances d'exécution

La commande [run-instances](#) suivante utilise l'idempotencie zonale car elle inclut un jeton client et une zone de disponibilité explicitement spécifiée.

```
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
```

Exemples de demandes d'API

Pour rendre une requête d'API idempotente, ajoutez le ClientToken paramètre.

Exemple 1 : Idempotencie

La demande [AllocateHosts](#)d'API suivante utilise l'idempotencie car elle inclut un jeton client.

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

Exemple 2 : RunInstances idempuissance régionale

La demande [RunInstances](#)d'API suivante utilise l'idempotencie régionale car elle inclut un jeton client mais ne spécifie pas explicitement ou implicitement de zone de disponibilité.

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

Exemple 3 : idempotence RunInstances zonale

La demande [RunInstances](#)d'API suivante utilise l'idempotencie zonale car elle inclut un jeton client et une zone de disponibilité explicitement spécifiée.

```
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

```

Recommandations de nouvelle tentative pour les demandes idempotentes

La table suivante présente certaines réponses courantes que vous pouvez obtenir pour les demandes d'API idempotentes et fournit des recommandations pour réessayer.

Réponse	Recommandation	Commentaires
200 (OK)	Ne pas réessayer	La demande d'origine s'est terminée avec succès. Toutes les tentatives suivantes sont renvoyées avec succès.
Codes de réponse de la série 400 (erreurs du client)	Ne pas réessayer	<p>La demande présente un problème, parmi les suivants :</p> <ul style="list-style-type: none"> • Elle inclut un paramètre ou une combinaison de paramètres invalide. • Elle utilise une action ou une ressource pour laquelle vous n'avez pas les autorisations nécessaires. • Elle utilise une ressource qui est en train de changer d'état. <p>Si la demande concerne une ressource en train de changer d'état, il est possible que la nouvelle tentative aboutisse.</p>

Réponse	Recommandation	Commentaires
Codes de réponse de la série 500 (erreurs du serveur)	Réessayer	L'erreur est due à un problème AWS côté serveur et est généralement transitoire. Répétez la demande avec une stratégie de backoff appropriée.

Limitation des demandes pour l'API Amazon EC2

Amazon EC2 limite les demandes d'API EC2 pour chaque AWS compte par région. Nous faisons cela pour améliorer les performances du service et garantir une utilisation équitable pour tous les clients Amazon EC2. La régulation garantit que les appels à l'API Amazon EC2 ne dépassent pas les limites de demandes d'API maximales autorisées. Les appels d'API sont soumis aux limites de demandes, qu'ils proviennent de :

- Une application tierce
- Un outil de ligne de commande
- La console Amazon EC2

Si vous dépassez une limite de limitation de l'API, le code `RequestLimitExceeded` d'erreur s'affiche.

Table des matières

- [Comment l'étranglement est appliqué](#)
- [Limites d'étranglement](#)
- [Surveiller la régulation des API](#)
- [Réessais et recul exponentiel](#)
- [Demander une augmentation de limite](#)

Comment l'étranglement est appliqué

Amazon EC2 utilise l'[algorithme Token Bucket](#) pour implémenter la régulation des API. Avec cet algorithme, votre compte dispose d'un compartiment contenant un nombre spécifique de jetons. Le nombre de jetons dans le compartiment représente votre limite d'étranglement à chaque seconde.

Amazon EC2 implémente deux types de régulation des API :

types de régulation de l'API

- [Limitation du débit de demande](#)
- [Limitation du débit des ressources](#)

Limitation du débit de demande

Avec la limitation du taux de demandes, le nombre de demandes d'API que vous effectuez est limité. Chaque requête que vous effectuez supprime un jeton du compartiment. Par exemple, la taille du compartiment pour les actions d'API non mutantes (`Describe*`) est de 100 jetons, vous pouvez donc effectuer jusqu'à 100 `Describe*` requêtes en une seconde. Si vous dépassez 100 demandes par seconde, vous êtes limité et les demandes restantes échouent au cours de cette seconde.

Les seaux se rechargent automatiquement à un débit défini. Si le compartiment est inférieur à sa capacité maximale, un nombre défini de jetons y est ajouté chaque seconde jusqu'à ce qu'il atteigne sa capacité maximale. Si le seau est plein à l'arrivée des jetons de recharge, ils sont jetés. Le bucket ne peut pas contenir plus de jetons que son maximum. Par exemple, la taille du compartiment pour les actions d'API non mutantes (`Describe*`) est de 100 jetons et le taux de recharge est de 20 jetons par seconde. Si vous effectuez 100 demandes d'`Describe*` API par seconde, le bucket est immédiatement réduit à zéro (0) jeton. Le seau est ensuite rempli de 20 jetons par seconde, jusqu'à ce qu'il atteigne sa capacité maximale de 100 jetons. Cela signifie que le seau précédemment vide atteint sa capacité maximale au bout de 5 secondes.

Il n'est pas nécessaire d'attendre que le compartiment soit complètement plein pour pouvoir effectuer des demandes d'API. Vous pouvez utiliser des jetons au fur et à mesure qu'ils sont ajoutés au bucket. Si vous utilisez immédiatement les jetons de recharge, le seau n'atteint pas sa capacité maximale. Par exemple, la taille du compartiment pour les actions non mutantes sur console est de 100 jetons, et le taux de recharge est de 10 jetons par seconde. Si vous épuisez le compartiment en effectuant 100 demandes d'API par seconde, vous pouvez continuer à effectuer 10 demandes d'API par seconde. Le compartiment ne peut être rempli à sa capacité maximale que si vous effectuez moins de 10 demandes d'API par seconde.

Limitation du débit des ressources

Certaines actions d'API, telles que `RunInstances` et `TerminateInstances`, comme décrit dans le tableau ci-dessous, utilisent la limitation du débit des ressources en plus de la limitation du débit des demandes. Ces actions d'API disposent d'un compartiment de jetons de ressources

distinct qui s'épuise en fonction du nombre de ressources affectées par la demande. À l'instar des compartiments de jetons de demande, les compartiments de jetons de ressources ont un nombre maximal de compartiments qui vous permet de les déborder, et un taux de recharge qui vous permet de maintenir un taux constant de demandes aussi longtemps que nécessaire. Si vous dépassez une limite de compartiment spécifique pour une API, y compris lorsqu'un compartiment n'est pas encore rempli pour prendre en charge le prochain appel d'API, l'action de l'API est limitée même si vous n'avez pas atteint la limite totale de l'API.

Par exemple, la taille du compartiment de jetons de ressources pour RunInstances est de 1 000 jetons et le taux de recharge est de deux jetons par seconde. Par conséquent, vous pouvez lancer immédiatement 1 000 instances en utilisant autant de demandes d'API que vous le souhaitez, par exemple une demande pour 1 000 instances ou quatre demandes pour 250 instances. Une fois le compartiment de jetons de ressources vide, vous pouvez lancer jusqu'à deux instances par seconde, en utilisant soit une demande pour deux instances, soit deux demandes pour une instance.

Pour plus d'informations, consultez [Tailles des seaux de jetons de ressources et taux de recharge](#).

Limites d'étranglement

Les sections suivantes décrivent les tailles des compartiments de jetons de demande et de jetons de ressources ainsi que les taux de recharge.

Limites

- [Demandez la taille des seaux de jetons et les taux de recharge](#)
- [Tailles des seaux de jetons de ressources et taux de recharge](#)

Demandez la taille des seaux de jetons et les taux de recharge

À des fins de limitation du taux de demandes, les actions de l'API sont regroupées dans les catégories suivantes :

- Actions non mutantes : actions d'API qui récupèrent des données sur les ressources. Cette catégorie inclut généralement toutes les Describe* actions, telles que DescribeRouteTablesDescribeImages, etDescribeHosts. Ces actions d'API présentent généralement les limites de limitation d'API les plus élevées.
- Actions non mutantes non filtrées et non paginées : sous-ensemble spécifique d'actions d'API non mutantes qui, lorsqu'elles sont appelées sans spécifier de [pagination](#) ni de [filtre](#), utilisent des jetons

provenant d'un compartiment de jetons plus petit. Il est recommandé d'utiliser la pagination et le filtrage afin que les jetons soient déduits du compartiment de jetons standard (plus grand).

- **Actions de mutation** : actions d'API qui créent, modifient ou suppriment des ressources. Cette catégorie inclut généralement toutes les actions d'API qui ne sont pas classées comme des actions non mutantes, telles que `CreateVolumeModifyHosts`, et `DeleteSnapshot`. La limite de régulation de ces actions est inférieure à celle des appels d'API sans mutation.
- **Actions gourmandes en ressources** : actions d'API mutantes qui prennent le plus de temps et consomment le plus de ressources. Ces actions ont une limite d'étranglement encore plus faible que les actions mutantes. Elles sont régulées séparément des autres actions mutantes.
- **Actions non mutantes de console** : actions d'API non mutantes appelées depuis la console Amazon EC2. Ces actions d'API sont limitées séparément des autres actions d'API non mutantes.
- **Actions non classées** : ces actions d'API reçoivent leur propre taille de bucket de jetons et leurs propres taux de recharge, même si, par définition, elles entrent dans l'une des autres catégories.

Le tableau suivant indique la taille des seaux de jetons demandés et les taux de recharge pour toutes les AWS régions.

Catégorie d'action de l'API	Actions	Capacité maximale du compartiment	Taux de recharge du seau
Actions non mutantes	<ul style="list-style-type: none"> • <code>Describe*</code> • <code>Get*</code> 	100	20
Actions non mutantes non filtrées et non paginées	<ul style="list-style-type: none"> • <code>DescribeInstances</code> • <code>DescribeNetworkInterfaces</code> • <code>DescribeVolumes</code> 	50	10

Catégorie d'action de l'API	Actions	Capacité maximale du compartiment	Taux de recharge du seau
	<ul style="list-style-type: none">DescribeInstanceStatusDescribeSnapshotsDescribeSecurityGroupsDescribeSpotInstanceRequests		
Actions mutantes	Actions d'API qui ne sont pas classées dans la catégorie des actions non mutantes.	200	5

Catégorie d'action de l'API	Actions	Capacité maximale du compartiment	Taux de recharge du seau
Actions gourmandes en ressources	<ul style="list-style-type: none"> • AuthorizeSecurityGroupIngress • CancelSpotInstanceRequests • CreateKeyPair • RequestSpotInstances • RevokeSecurityGroupIngress • CreateVpcPeeringConnection • AcceptVpcPeeringConnection • RejectVpcPeeringConnection • DeleteVpcPeeringConnection 	50	5

Catégorie d'action de l'API	Actions	Capacité maximale du compartiment	Taux de recharge du seau
Actions non mutantes de la console	<ul style="list-style-type: none"> Describe* Get* 	100	10
Actions non classées	RunInstances	5	2
	StartInstances	5	2
	CreateVpcEndpoint	4	0.3
	ModifyVpcEndpoint	4	0.3
	DeleteVpcEndpoints	4	0.3
	AcceptVpcEndpointConnections	10	1
	RejectVpcEndpointConnections	10	1
	CreateVpcEndpointServiceConfiguration	10	1
	ModifyVpcEndpointServiceConfiguration	10	1

Catégorie d'action de l'API	Actions	Capacité maximale du compartiment	Taux de recharge du seau
	DeleteVpc EndpointS erviceCon figurations	10	1
	CreateDef aultVpc	1	1
	CreateDef aultSubnet	1	1
	MoveAddre ssToVpc	1	1
	RestoreAd dressToClassic	1	1
	DescribeM ovingAddresses	1	1
	Advertise ByoipCidr	1	0.1
	Provision ByoipCidr	1	0.1
	DescribeB yoipCidrs	1	0.5
	Deprovisi onByoipCidr	1	0.1
	WithdrawB yoipCidr	1	0.1

Catégorie d'action de l'API	Actions	Capacité maximale du compartiment	Taux de recharge du seau
	DescribeReservedInstancesOfferings	10	10
	PurchaseReservedInstancesOffering	5	5
	DescribeSpotFleetRequests	50	3
	DescribeSpotFleetInstances	100	5
	DescribeSpotFleetRequestHistory	100	5
	AssociateEnclaveCertificateIamRole	10	1
	DisassociateEnclaveCertificateIamRole	10	1

Catégorie d'action de l'API	Actions	Capacité maximale du compartiment	Taux de recharge du seau
	GetAssociatedEnclaveCertificateIamRoles	10	1
	GetConsoleScreenshot	5 par compte 2 par instance	5 par compte 1 par instance

Tailles des seaux de jetons de ressources et taux de recharge

Le tableau suivant répertorie les tailles des compartiments de jetons de ressources et les taux de recharge pour les actions d'API qui utilisent la limitation du débit des ressources.

Action d'API	Capacité maximale du compartiment	Taux de recharge du seau
RunInstances	1 000	2
TerminateInstances	1 000	20
StartInstances	1 000	2
StopInstances	1 000	20

Surveiller la régulation des API

Vous pouvez utiliser Amazon CloudWatch pour surveiller vos appels d'API Amazon EC2 et pour collecter et suivre les indicateurs relatifs à la limitation des API. Vous pouvez également créer une alarme pour vous avertir lorsque vous êtes sur le point d'atteindre les limites de limitation de l'API. Pour plus d'informations, consultez [Surveillez les demandes d'API Amazon EC2 à l'aide d'Amazon CloudWatch](#).

Réessais et recul exponentiel

Il se peut que votre application doive réessayer une demande d'API. Par exemple :

- Pour vérifier l'existence d'une mise à jour du statut d'une ressource
- Pour énumérer un grand nombre de ressources (par exemple, tous vos volumes)
- Pour réessayer une demande après qu'elle ait échoué en raison d'une erreur de serveur (5xx) ou d'une erreur de limitation

Toutefois, en cas d'erreur client (4xx), vous devez réviser la demande pour corriger le problème avant de réessayer la demande.

Changements de statut des ressources

Avant de commencer à interroger pour vérifier les mises à jour de statut, donnez à la demande le temps de se terminer éventuellement. Par exemple, attendez quelques minutes avant de vérifier si votre instance est active. Lorsque vous commencez à interroger, utilisez un intervalle de sommeil approprié entre les demandes successives afin de réduire le taux de demandes d'API. Pour obtenir de meilleurs résultats, utilisez un intervalle de veille croissant ou variable.

Vous pouvez également utiliser Amazon EventBridge pour vous informer de l'état de certaines ressources. Par exemple, vous pouvez utiliser l'événement de notification de changement d'état d'une instance EC2 pour vous informer d'un changement d'état d'une instance. Pour plus d'informations, consultez [Automatiser Amazon EC2](#) à l'aide de EventBridge

Nouvelle tentative

Lorsque vous devez interroger ou réessayer une demande d'API, nous vous recommandons d'utiliser un algorithme de ralentissement exponentiel pour calculer l'intervalle de sommeil entre les appels d'API. L'idée sous-jacente consiste à utiliser des temps d'attente progressivement plus longs entre les tentatives en cas de réponses d'erreur consécutives. Vous devez implémenter un intervalle maximal, ainsi qu'un nombre maximal de nouvelles tentatives. Vous pouvez également utiliser la gigue (délai aléatoire) pour éviter les collisions successives. Pour plus d'informations, consultez les [sections Expiration, nouvelles tentatives et interruption avec gigue](#).

Chaque AWS SDK implémente une logique de nouvelle tentative automatique. Pour plus d'informations, consultez la section [Comportement des tentatives](#) dans le Guide de référence AWS des SDK et des outils.

Demander une augmentation de limite

Vous pouvez demander une augmentation des limites de limitation des API pour votre. Compte AWS

Pour demander l'accès à cette fonctionnalité

1. [AWS Support Centre](#) ouvert.
2. Choisissez Create case (Créer une demande).
3. Choisissez Compte et facturation.
4. Pour le service, sélectionnez Informations générales et Mise en route.
5. Dans Catégorie, choisissez Utilisation AWS et services.
6. Choisissez Next step: Additional information (Étape suivante : informations supplémentaires).
7. Pour Subject (Objet), saisissez **Request an increase in my Amazon EC2 API throttling limits**.
8. Pour Description, saisissez **Please increase the API throttling limits for my account. Related page: <https://docs.aws.amazon.com/AWSEC2/latest/APIReference/throttling.html>**. Incluez également les informations suivantes :
 - Description de votre cas d'utilisation.
 - Les régions où vous avez besoin d'une augmentation.
 - La fenêtre d'une heure, en UTC, au cours de laquelle le pic d'étranglement ou d'utilisation s'est produit (pour calculer la nouvelle limite d'étranglement).
9. Choisissez Next step: Solve now or contact us (Étape suivante : résolvez maintenant ou contactez-nous).
10. Dans l'onglet Contactez-nous, choisissez la langue de contact et le mode de contact que vous préférez.
11. Sélectionnez Envoyer.

Créez des ressources Amazon EC2 à l'aide du AWS CLI

Vous pouvez créer et gérer vos ressources Amazon EC2 à l'aide du AWS Command Line Interface (AWS CLI) dans un shell de ligne de commande. AWS CLI Fournit un accès direct aux API pour AWS services, par exemple, Amazon EC2.

Pour obtenir la syntaxe et des exemples de commandes pour Amazon EC2, consultez [ec2](#) dans le AWS CLI manuel Command Reference. Vous pouvez également trouver ces exemples dans [aws-cli/awscli/examples/ec2](#) sur github.

En savoir plus sur le AWS CLI

Pour en savoir plus à ce sujet AWS CLI, consultez les ressources suivantes :

- [AWS Command Line Interface](#)
- [AWS Command Line Interface Guide de l'utilisateur pour la version 2](#)
- [AWS Command Line Interface Guide de l'utilisateur pour la version 1](#)

Créez des ressources Amazon EC2 à l'aide de AWS CloudFormation

Amazon EC2 est intégré à un service qui vous aide à modéliser et à configurer vos AWS ressources afin que vous puissiez passer moins de temps à créer et à gérer vos ressources et votre infrastructure. AWS CloudFormation Vous créez un modèle qui décrit les AWS ressources dont vous avez besoin (telles que les instances et les sous-réseaux), puis vous AWS CloudFormation approvisionnez et configurez ces ressources pour vous.

Lorsque vous l'utilisez AWS CloudFormation, vous pouvez réutiliser votre modèle pour configurer vos ressources Amazon EC2 de manière cohérente et répétée. Décrivez vos ressources une seule fois, puis distribuez les mêmes ressources encore et encore dans plusieurs Comptes AWS régions.

Amazon EC2 et modèles AWS CloudFormation

[Pour fournir et configurer des ressources pour Amazon EC2 et les services associés, vous devez comprendre AWS CloudFormation les modèles.](#) Les modèles sont des fichiers texte formatés en JSON ou YAML. Ces modèles décrivent les ressources que vous allez mettre à disposition dans vos AWS CloudFormation piles. Si vous n'êtes pas familiarisé avec JSON ou YAML, vous pouvez utiliser AWS CloudFormation Designer pour vous aider à démarrer avec les AWS CloudFormation modèles. Pour plus d'informations, voir [Qu'est-ce que AWS CloudFormation Designer ?](#) dans le guide de AWS CloudFormation l'utilisateur.

Ressources pour Amazon EC2

Ressources informatiques

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

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

Ressources de mise en réseau

- [AWS::EC2::CarrierGateway](#)
- [AWS::EC2::ClientVpnAuthorizationRule](#)
- [AWS::EC2::ClientVpnPoint final](#)
- [AWS::EC2::ClientVpnParcours](#)
- [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::LocalGatewayParcours](#)
- [AWS::EC2::LocalGatewayRouteTable](#)
- [AWS::EC2::LocalGatewayRouteTableVirtualInterfaceGroupAssociation](#)
- [AWS::EC2::LocalGatewayRouteTableAssociation VPC](#)
- [AWS::EC2::NatGateway](#)

- [AWS::EC2::NetworkInterface](#)
- [AWS::EC2::NetworkInsightsAccessScope](#)
- [AWS::EC2::NetworkInsightsAccessScopeAnalysis](#)
- [AWS::EC2::NetworkInsightsAnalyse](#)
- [AWS::EC2::NetworkInsightsParcours](#)
- [AWS::EC2::NetworkInterfacePièce jointe](#)
- [AWS::EC2::NetworkInterfaceAutorisation](#)
- [AWS::EC2::NetworkPerformanceMetricSubscription](#)
- [AWS::EC2::PrefixList](#)
- [AWS::EC2::Route](#)
- [AWS::EC2::RouteTable](#)
- [AWS::EC2::Subnet](#)
- [AWS::EC2::SubnetCidrBloquer](#)
- [AWS::EC2::SubnetNetworkAclAssociation](#)
- [AWS::EC2::SubnetRouteTableAssociation](#)
- [AWS::EC2::TrafficMirrorFiltre](#)
- [AWS::EC2::TrafficMirrorFilterRule](#)
- [AWS::EC2::TrafficMirrorSéance](#)
- [AWS::EC2::TrafficMirrorCible](#)
- [AWS::EC2::TransitGateway](#)
- [AWS::EC2::TransitGatewayPièce jointe](#)
- [AWS::EC2::TransitGatewayConnect](#)
- [AWS::EC2::TransitGatewayMulticastDomain](#)
- [AWS::EC2::TransitGatewayMulticastDomainAssociation](#)
- [AWS::EC2::TransitGatewayMulticastGroupMember](#)
- [AWS::EC2::TransitGatewayMulticastGroupSource](#)
- [AWS::EC2::TransitGatewayPeeringAttachment](#)
- [AWS::EC2::TransitGatewayParcours](#)
- [AWS::EC2::TransitGatewayRouteTable](#)
- [AWS::EC2::TransitGatewayRouteTableAssociation](#)

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

Ressources de sécurité

- [AWS::EC2::KeyPair](#)
- [AWS::EC2::NetworkAcl](#)
- [AWS::EC2::NetworkAclEntrée](#)
- [AWS::EC2::SecurityGroup](#)
- [AWS::EC2::SecurityGroupSortie](#)
- [AWS::EC2::SecurityGroupEntrée](#)
- [AWS::EC2::VerifiedAccessPoint final](#)
- [AWS::EC2::VerifiedAccessGroupe](#)
- [AWS::EC2::VerifiedAccessInstance](#)
- [AWS::EC2::VerifiedAccessTrustProvider](#)

Ressources de stockage

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

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

En savoir plus sur AWS CloudFormation

Pour en savoir plus AWS CloudFormation, consultez les ressources suivantes :

- [AWS CloudFormation](#)
- [AWS CloudFormation Guide de l'utilisateur](#)

Création de ressources Amazon EC2 à l'aide d'un SDK AWS

AWS fournit des kits de développement logiciel (SDK) pour de nombreux langages de programmation populaires. Un SDK rend le développement plus efficace en fournissant les éléments suivants :

- Composants et bibliothèques prédéfinis que vous pouvez intégrer à vos applications
- Outils spécifiques au langage, tels que les compilateurs et les débogueurs
- Signature cryptographique des demandes de service
- Demander de nouvelles tentatives
- Gestion des réponses aux erreurs

Exemples de code pour l'API Amazon EC2

Les exemples de code fournis par vous AWS montrent comment utiliser une API et accomplir des tâches spécifiques. Pour des exemples relatifs à l'API Amazon EC2, consultez la section [Exemples de code pour Amazon EC2](#). Pour des exemples supplémentaires, consultez [Rechercher des exemples de code pour les AWS SDK](#) ou [aws-doc-sdk-examples](#) sur github.

En savoir plus sur les AWS SDK

Pour en savoir plus sur les AWS SDK, consultez les ressources suivantes :

- [AWS Guide de référence des SDK et des outils](#)
- [Des outils sur lesquels s'appuyer AWS](#)
- [Qu'est-ce qu'un SDK ?](#)

API de bas niveau pour Amazon EC2

L'API de bas niveau pour Amazon EC2 est l'interface de niveau protocole pour Amazon EC2.

Lorsque vous utilisez l'API de bas niveau, vous devez formater correctement chaque requête HTTPS et ajouter une signature numérique valide à chaque demande. Pour plus d'informations, consultez la section [Envoyer des demandes à l'API Amazon EC2 dans le manuel Amazon EC2 API Reference](#).

Vous pouvez également utiliser un AWS SDK, qui construit et signe les demandes en votre nom.

Pour plus d'informations, consultez [Utilisation d'un AWS SDK](#).

L'API Amazon EC2 comprend des actions et des types de données pour plusieurs services. Pour consulter les actions de chaque service, consultez les pages suivantes du manuel Amazon EC2 API Reference.

- [AWS Client VPN actions](#)
- [Actions Amazon EBS](#)
- [Actions Amazon EC2](#)
- [AWS Network Manager actions](#)
- [AWS Actions de Nitro Enclaves](#)
- [AWS Outposts actions](#)
- [AWS PrivateLink actions](#)
- [Actions relatives à la corbeille](#)
- [AWS Site-to-Site VPN actions](#)
- [AWS Transit Gateway actions](#)
- [Accès vérifié par AWS actions](#)
- [Actions d'importation/exportation de machines virtuelles](#)
- [Actions Amazon VPC](#)
- [Actions IPAM d'Amazon VPC](#)
- [AWS Wavelength actions](#)

Génération de code pour les actions de votre console à l'aide de Console-to-Code

Console-to-Code est en version préliminaire pour Amazon EC2 et est susceptible d'être modifié. Uniquement disponible dans la région USA Est (Virginie du Nord)

La console fournit un parcours guidé pour créer des ressources et tester des prototypes. Si vous souhaitez créer les mêmes ressources à grande échelle, vous aurez besoin d'un code d'automatisation. Console-to-Code est une fonctionnalité de la console Amazon EC2 qui peut vous aider à démarrer avec votre code d'automatisation. Console-to-Code enregistre les actions de votre console, y compris les valeurs par défaut et les paramètres compatibles. Il utilise ensuite l'IA générative pour suggérer du code dans votre format préféré infrastructure-as-code (iAc) pour les actions que vous souhaitez. Vous pouvez utiliser le code comme point de départ, en le personnalisant pour qu'il soit prêt pour la production en fonction de votre cas d'utilisation spécifique.

L'utilisation de Console-to-Code n'entraîne aucun coût supplémentaire.

Comment ça marche

Console-to-Code peut vous aider à faire vos premiers avec le code d'automatisation comme suit :

1. Vous effectuez des actions dans la console, telles que le lancement d'une instance ou l'activation d'une surveillance détaillée.
2. Console-to-Code enregistre toutes vos actions, y compris tous les paramètres par défaut et compatibles fournis par la console.
3. Vous choisissez les actions que vous souhaitez utiliser dans vos scripts d'automatisation. Il peut s'agir d'actions de mutation ou de lecture seule (non-mutation), ou des deux types d'action.
4. Console-to-Code génère du code dans le format de votre choix infrastructure-as-code (iAc), par exemple. TypeScript
5. Vous copiez le code pour l'utiliser dans votre outil de développement de code ou vous le téléchargez pour le partager.
6. Utilisez ces requêtes comme point de départ pour vos propres rapports. Vous devrez vérifier que le code répond à vos attentes et que les paramètres configureront vos ressources comme prévu.

Vous devrez personnaliser le code pour qu'il soit prêt pour la production en fonction de votre cas d'utilisation. Une fois que vous êtes satisfait du code, vous pouvez l'utiliser dans vos scripts d'automatisation.

Pour savoir comment utiliser Console-to-Code dans la console Amazon EC2, consultez [Utilisation de Console-to-Code](#).

Limites

Les limitations suivantes s'appliquent lors de l'utilisation de Console-to-Code.

Régions prises en charge

Uniquement disponible actuellement dans la région USA Est (Virginie du Nord).

Formats de code pris en charge

La console vers le code peut actuellement générer infrastructure-as-code (iAc) dans les formats de code suivants :

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

Actions conservées

- Session en cours : seules les actions effectuées pendant la session en cours sont affichées dans le tableau Actions enregistrées. Les actions entreprises lors des sessions précédentes ne sont pas conservées.
- Actualisation du navigateur : les actions enregistrées sont perdues lorsque vous actualisez l'onglet du navigateur.
- Isolation des onglets : le tableau des actions enregistrées est spécifique à l'onglet du navigateur dans lequel les actions ont été effectuées. Les actions effectuées dans un onglet ne sont pas visibles dans le tableau Actions enregistrées d'un autre onglet.

Tableau Actions enregistrées

Le tableau suivant répertorie et décrit les colonnes du tableau Actions enregistrées dans la console Console-to-Code.

Titre de colonne	Description
Page de console	La page de console sur laquelle l'action a été effectuée.
Opération	L'opération d'API.
Type	Type d'action. <ul style="list-style-type: none">• Mutations : actions d'API qui créent, modifient ou suppriment des ressources.• Lecture seule : actions d'API qui récupèrent des données sur les ressources (généralement toutes les actions Describe*).
Commande de la CLI	Détails sur l'action entreprise, y compris les paramètres et les valeurs.
Heure de création	Heure à laquelle l'action a été entreprise.

Utilisation de Console-to-Code

Suivez les instructions ci-dessous pour générer du code à l'aide de Console-to-Code dans la console Amazon EC2.

Pour voir une animation de ces étapes, consultez [Afficher une animation : génération de code à l'aide de Console-to-Code dans la console Amazon EC2](#).

Pour générer du code à l'aide de Console-to-Code

1. [Ouvrez la console Amazon EC2 dans la région USA Est \(Virginie du Nord\) à l'adresse https://console.aws.amazon.com/ec2/home?region=us-east-1](https://console.aws.amazon.com/ec2/home?region=us-east-1).

 Note

Console-to-Code est en version préliminaire et n'est actuellement disponible que dans la région USA Est (Virginie du Nord). Seules les actions effectuées dans cette région seront enregistrées.

2. Utilisez la console pour créer des ressources et tester des prototypes. Par exemple, utilisez la console pour configurer et lancer des instances et activer une surveillance détaillée.

Console-to-Code enregistre toutes les actions que vous effectuez.

3. Dans le volet de navigation de gauche, choisissez Console-to-Code.
4. Dans le tableau Actions enregistrées, examinez les actions enregistrées et déterminez les actions à inclure pour la génération de code.
 - Utilisez le champ de recherche pour filtrer le tableau en fonction d'une page ou d'une action de console spécifique. Lorsque vous commencez la saisie, le tableau est filtré.
 - Utilisez le menu déroulant Type pour filtrer en fonction de toutes les actions, des actions de mutation ou des actions en lecture seule.

 Note

Seules les actions entreprises au cours de la session en cours sont répertoriées. Pour plus d'informations, consultez [Actions conservées](#).

5. Cochez la case en regard de chaque action pour laquelle vous souhaitez que le code soit généré.

 Note

Jusqu'à 5 actions peuvent être sélectionnées à la fois.

6. Cliquez sur le bouton Générer le code {code}.

L'étiquette du bouton utilise par défaut le dernier format de code sélectionné. Pour sélectionner un autre format de code, cliquez sur la flèche située à côté du bouton.

7. Sous Réviser le code, choisissez Copier pour copier le code à utiliser dans votre outil de développement ou Télécharger pour télécharger le fichier à partager.
8. Utilisez le code comme point de départ pour votre infrastructure-as-code. Vous devrez personnaliser le code pour qu'il soit prêt pour la production en fonction de votre cas d'utilisation spécifique.

 Note

Si vous constatez que le code n'est pas prêt pour la production, veuillez nous faire part de vos commentaires sur la manière dont il peut être amélioré (voir l'étape 9 suivante). AWS Support ne peut pas vous aider avec le code généré ou le développement de votre code personnalisé.

9. (Facultatif) Choisissez le pouce levé ou le pouce vers le bas pour nous faire savoir si Console-to-Code vous a été utile. Si vous choisissez le pouce vers le bas, vous pouvez ensuite sélectionner Envoyer des commentaires pour nous indiquer comment nous pouvons améliorer le code afin de mieux vous aider.

Afficher une animation : génération de code à l'aide de Console-to-Code dans la console Amazon EC2

The screenshot displays the Amazon EC2 console interface. On the left is a navigation sidebar with categories like 'Instances', 'Images', 'Elastic Block Store', and 'Network & Security'. The main content area is divided into several panels:

- Resources:** A table showing the number of EC2 resources in the US East (N. Virginia) Region.

Resources	
You are using the following Amazon EC2 resources in the US East (N. Virginia) Region:	
Instances (running)	4
Dedicated Hosts	0
Instances	5
Load balancers	0
Security groups	14
Volumes	6
Auto Scaling Groups	0
Elastic IPs	0
Key pairs	5
Placement groups	1
Snapshots	5
- Launch instance:** A panel with a 'Launch instance' button and a 'Migrate a server' button. It includes a note: 'Note: Your Instances will launch in the US East (N. Virginia) Region'.
- Service health:** Shows the 'AWS Health Dashboard' and the current region 'US East (N. Virginia)'. It also lists available zones:

Zone name	Zone ID
us-east-1a	use1-az2
- Account attributes:** Displays account information such as 'Default VPC' (vpc-92304aeb) and various settings links.
- Explore AWS:** Promotional banners for 'Save up to 90% on EC2 with Spot Instances', 'Amazon GuardDuty Malware Protection', and 'Enable Best Price-Performance with AWS Graviton2'.

Exemples de code pour Amazon EC2 à l'aide de kits SDK AWS

Les exemples de code suivants montrent comment utiliser Amazon EC2 avec un kit de développement AWS logiciel (SDK).

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Mise en route

Hello Amazon EC2

Les exemples de code suivants montrent comment démarrer avec Amazon EC2.

.NET

AWS SDK for .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
namespace EC2Actions;

public class HelloEc2
{
```

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

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

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

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

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

- Pour plus de détails sur l'API, voir [DescribeSecurityGroups](#) la section Référence des AWS SDK for .NET API.

C++

SDK pour C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Code pour le MakeLists fichier CMake C.txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS ec2)

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

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
  libraries for the AWS SDK.
  string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
    "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
  list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
  # Copy relevant AWS SDK for C++ libraries into the current binary directory
  for running and debugging.
```

```
# set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
may need to uncomment this

                                # and set the proper subdirectory to the
executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
    ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
    hello_ec2.cpp)

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

Code pour le fichier source hello_ec2.cpp.

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

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

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
```

```
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

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

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

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

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

                Aws::String typeString =

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

                Aws::String monitorString =

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

```

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

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

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

```

- Pour plus de détails sur l'API, voir [DescribeSecurityGroups](#) la section Référence des AWS SDK for C++ API.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
    try {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupIds(groupId)
            .build();

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

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

- Pour plus de détails sur l'API, voir [DescribeSecurityGroups](#) la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

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

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

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

- Pour plus de détails sur l'API, voir [DescribeSecurityGroups](#) la section Référence des AWS SDK for JavaScript API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

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

- Pour plus de détails sur l'API, consultez [DescribeSecurityGroups](#) la section AWS SDK pour la référence de l'API Kotlin.

Python

SDK pour Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import boto3

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

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

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

- Pour plus de détails sur l'API, consultez [DescribeSecurityGroups](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Exemples de code

- [Actions pour Amazon EC2 à l'aide de kits SDK AWS](#)
 - [Utilisation AcceptVpcPeeringConnection avec un AWS SDK ou une CLI](#)
 - [Utilisation AllocateAddress avec un AWS SDK ou une CLI](#)
 - [Utilisation AllocateHosts avec un AWS SDK ou une CLI](#)
 - [Utilisation AssignPrivateIpAddresses avec un AWS SDK ou une CLI](#)
 - [Utilisation AssociateAddress avec un AWS SDK ou une CLI](#)
 - [Utilisation AssociateDhcpOptions avec un AWS SDK ou une CLI](#)
 - [Utilisation AssociateRouteTable avec un AWS SDK ou une CLI](#)
 - [Utilisation AttachInternetGateway avec un AWS SDK ou une CLI](#)
 - [Utilisation AttachNetworkInterface avec un AWS SDK ou une CLI](#)

- [Utilisation AttachVolume avec un AWS SDK ou une CLI](#)
- [Utilisation AttachVpnGateway avec un AWS SDK ou une CLI](#)
- [Utilisation AuthorizeSecurityGroupEgress avec un AWS SDK ou une CLI](#)
- [Utilisation AuthorizeSecurityGroupIngress avec un AWS SDK ou une CLI](#)
- [Utilisation CancelCapacityReservation avec un AWS SDK ou une CLI](#)
- [Utilisation CancelImportTask avec un AWS SDK ou une CLI](#)
- [Utilisation CancelSpotFleetRequests avec un AWS SDK ou une CLI](#)
- [Utilisation CancelSpotInstanceRequests avec un AWS SDK ou une CLI](#)
- [Utilisation ConfirmProductInstance avec un AWS SDK ou une CLI](#)
- [Utilisation CopyImage avec un AWS SDK ou une CLI](#)
- [Utilisation CopySnapshot avec un AWS SDK ou une CLI](#)
- [Utilisation CreateCapacityReservation avec un AWS SDK ou une CLI](#)
- [Utilisation CreateCustomerGateway avec un AWS SDK ou une CLI](#)
- [Utilisation CreateDhcpOptions avec un AWS SDK ou une CLI](#)
- [Utilisation CreateFlowLogs avec un AWS SDK ou une CLI](#)
- [Utilisation CreateImage avec un AWS SDK ou une CLI](#)
- [Utilisation CreateInstanceExportTask avec un AWS SDK ou une CLI](#)
- [Utilisation CreateInternetGateway avec un AWS SDK ou une CLI](#)
- [Utilisation CreateKeyPair avec un AWS SDK ou une CLI](#)
- [Utilisation CreateLaunchTemplate avec un AWS SDK ou une CLI](#)
- [Utilisation CreateNetworkAcl avec un AWS SDK ou une CLI](#)
- [Utilisation CreateNetworkAclEntry avec un AWS SDK ou une CLI](#)
- [Utilisation CreateNetworkInterface avec un AWS SDK ou une CLI](#)
- [Utilisation CreatePlacementGroup avec un AWS SDK ou une CLI](#)
- [Utilisation CreateRoute avec un AWS SDK ou une CLI](#)
- [Utilisation CreateRouteTable avec un AWS SDK ou une CLI](#)
- [Utilisation CreateSecurityGroup avec un AWS SDK ou une CLI](#)
- [Utilisation CreateSnapshot avec un AWS SDK ou une CLI](#)
- [Utilisation CreateSpotDatafeedSubscription avec un AWS SDK ou une CLI](#)
- [Utilisation CreateSubnet avec un AWS SDK ou une CLI](#)

- [Utilisation CreateTags avec un AWS SDK ou une CLI](#)
- [Utilisation CreateVolume avec un AWS SDK ou une CLI](#)
- [Utilisation CreateVpc avec un AWS SDK ou une CLI](#)
- [Utilisation CreateVpcEndpoint avec un AWS SDK ou une CLI](#)
- [Utilisation CreateVpnConnection avec un AWS SDK ou une CLI](#)
- [Utilisation CreateVpnConnectionRoute avec un AWS SDK ou une CLI](#)
- [Utilisation CreateVpnGateway avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteCustomerGateway avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteDhcpOptions avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteFlowLogs avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteInternetGateway avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteKeyPair avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteLaunchTemplate avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteNetworkAcl avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteNetworkAclEntry avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteNetworkInterface avec un AWS SDK ou une CLI](#)
- [Utilisation DeletePlacementGroup avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteRoute avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteRouteTable avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteSecurityGroup avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteSnapshot avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteSpotDatafeedSubscription avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteSubnet avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteTags avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteVolume avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteVpc avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteVpnConnection avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteVpnConnectionRoute avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteVpnGateway avec un AWS SDK ou une CLI](#)
- [Utilisation DeregisterImage avec un AWS SDK ou une CLI](#)

- [Utilisation DescribeAccountAttributes avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeAddresses avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeAvailabilityZones avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeBundleTasks avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeCapacityReservations avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeCustomerGateways avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeDhcpOptions avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeFlowLogs avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeHostReservationOfferings avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeHosts avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeIamInstanceProfileAssociations avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeIdFormat avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeIdentityIdFormat avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeImageAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeImages avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeImportImageTasks avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeImportSnapshotTasks avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeInstanceAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeInstanceStatus avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeInstanceTypes avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeInstances avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeInternetGateways avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeKeyPairs avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeNetworkAcls avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeNetworkInterfaceAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeNetworkInterfaces avec un AWS SDK ou une CLI](#)
- [Utilisation DescribePlacementGroups avec un AWS SDK ou une CLI](#)
- [Utilisation DescribePrefixLists avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeRegions avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeRouteTables avec un AWS SDK ou une CLI](#)

- [Utilisation DescribeScheduledInstanceAvailability avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeScheduledInstances avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeSecurityGroups avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeSnapshotAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeSnapshots avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeSpotDatafeedSubscription avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeSpotFleetInstances avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeSpotFleetRequestHistory avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeSpotFleetRequests avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeSpotInstanceRequests avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeSpotPriceHistory avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeSubnets avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeTags avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeVolumeAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeVolumeStatus avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeVolumes avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeVpcAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeVpcClassicLink avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeVpcClassicLinkDnsSupport avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeVpcEndpointServices avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeVpcEndpoints avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeVpcs avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeVpnConnections avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeVpnGateways avec un AWS SDK ou une CLI](#)
- [Utilisation DetachInternetGateway avec un AWS SDK ou une CLI](#)
- [Utilisation DetachNetworkInterface avec un AWS SDK ou une CLI](#)
- [Utilisation DetachVolume avec un AWS SDK ou une CLI](#)
- [Utilisation DetachVpnGateway avec un AWS SDK ou une CLI](#)
- [Utilisation DisableVgwRoutePropagation avec un AWS SDK ou une CLI](#)
- [Utilisation DisableVpcClassicLink avec un AWS SDK ou une CLI](#)

- [Utilisation DisableVpcClassicLinkDnsSupport avec un AWS SDK ou une CLI](#)
- [Utilisation DisassociateAddress avec un AWS SDK ou une CLI](#)
- [Utilisation DisassociateRouteTable avec un AWS SDK ou une CLI](#)
- [Utilisation EnableVgwRoutePropagation avec un AWS SDK ou une CLI](#)
- [Utilisation EnableVolumelo avec un AWS SDK ou une CLI](#)
- [Utilisation EnableVpcClassicLink avec un AWS SDK ou une CLI](#)
- [Utilisation EnableVpcClassicLinkDnsSupport avec un AWS SDK ou une CLI](#)
- [Utilisation GetConsoleOutput avec un AWS SDK ou une CLI](#)
- [Utilisation GetHostReservationPurchasePreview avec un AWS SDK ou une CLI](#)
- [Utilisation GetPasswordData avec un AWS SDK ou une CLI](#)
- [Utilisation ImportImage avec un AWS SDK ou une CLI](#)
- [Utilisation ImportKeyPair avec un AWS SDK ou une CLI](#)
- [Utilisation ImportSnapshot avec un AWS SDK ou une CLI](#)
- [Utilisation ModifyCapacityReservation avec un AWS SDK ou une CLI](#)
- [Utilisation ModifyHosts avec un AWS SDK ou une CLI](#)
- [Utilisation ModifyIdFormat avec un AWS SDK ou une CLI](#)
- [Utilisation ModifyImageAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation ModifyInstanceAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation ModifyInstanceCreditSpecification avec un AWS SDK ou une CLI](#)
- [Utilisation ModifyNetworkInterfaceAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation ModifyReservedInstances avec un AWS SDK ou une CLI](#)
- [Utilisation ModifySnapshotAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation ModifySpotFleetRequest avec un AWS SDK ou une CLI](#)
- [Utilisation ModifySubnetAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation ModifyVolumeAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation ModifyVpcAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation MonitorInstances avec un AWS SDK ou une CLI](#)
- [Utilisation MoveAddressToVpc avec un AWS SDK ou une CLI](#)
- [Utilisation PurchaseHostReservation avec un AWS SDK ou une CLI](#)
- [Utilisation PurchaseScheduledInstances avec un AWS SDK ou une CLI](#)

- [Utilisation RebootInstances avec un AWS SDK ou une CLI](#)
- [Utilisation RegisterImage avec un AWS SDK ou une CLI](#)
- [Utilisation RejectVpcPeeringConnection avec un AWS SDK ou une CLI](#)
- [Utilisation ReleaseAddress avec un AWS SDK ou une CLI](#)
- [Utilisation ReleaseHosts avec un AWS SDK ou une CLI](#)
- [Utilisation ReplacelamInstanceProfileAssociation avec un AWS SDK ou une CLI](#)
- [Utilisation ReplaceNetworkAclAssociation avec un AWS SDK ou une CLI](#)
- [Utilisation ReplaceNetworkAclEntry avec un AWS SDK ou une CLI](#)
- [Utilisation ReplaceRoute avec un AWS SDK ou une CLI](#)
- [Utilisation ReplaceRouteTableAssociation avec un AWS SDK ou une CLI](#)
- [Utilisation ReportInstanceStatus avec un AWS SDK ou une CLI](#)
- [Utilisation RequestSpotFleet avec un AWS SDK ou une CLI](#)
- [Utilisation RequestSpotInstances avec un AWS SDK ou une CLI](#)
- [Utilisation ResetImageAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation ResetInstanceAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation ResetNetworkInterfaceAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation ResetSnapshotAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation RevokeSecurityGroupEgress avec un AWS SDK ou une CLI](#)
- [Utilisation RevokeSecurityGroupIngress avec un AWS SDK ou une CLI](#)
- [Utilisation RunInstances avec un AWS SDK ou une CLI](#)
- [Utilisation RunScheduledInstances avec un AWS SDK ou une CLI](#)
- [Utilisation StartInstances avec un AWS SDK ou une CLI](#)
- [Utilisation StopInstances avec un AWS SDK ou une CLI](#)
- [Utilisation TerminateInstances avec un AWS SDK ou une CLI](#)
- [Utilisation UnassignPrivateIpAddresses avec un AWS SDK ou une CLI](#)
- [Utilisation UnmonitorInstances avec un AWS SDK ou une CLI](#)
- [Scénarios pour Amazon EC2 utilisant des kits de développement logiciel AWS](#)
 - [Créez et gérez un service résilient à l'aide d'un AWS SDK](#)
 - [Commencez à utiliser les instances Amazon EC2 à l'aide d'un SDK AWS](#)

Actions pour Amazon EC2 à l'aide de kits SDK AWS

Les exemples de code suivants montrent comment effectuer des actions Amazon EC2 individuelles à l'aide AWS de kits SDK. Ces extraits appellent l'API Amazon EC2 et sont des extraits de code de programmes plus volumineux qui doivent être exécutés en contexte. Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions pour configurer et exécuter le code.

Les exemples suivants incluent uniquement les actions les plus couramment utilisées. Pour obtenir la liste complète, consultez la section [Référence d'API Amazon Elastic Compute Cloud \(Amazon EC2\)](#) (français non garanti).

Exemples

- [Utilisation AcceptVpcPeeringConnection avec un AWS SDK ou une CLI](#)
- [Utilisation AllocateAddress avec un AWS SDK ou une CLI](#)
- [Utilisation AllocateHosts avec un AWS SDK ou une CLI](#)
- [Utilisation AssignPrivateIpAddresses avec un AWS SDK ou une CLI](#)
- [Utilisation AssociateAddress avec un AWS SDK ou une CLI](#)
- [Utilisation AssociateDhcpOptions avec un AWS SDK ou une CLI](#)
- [Utilisation AssociateRouteTable avec un AWS SDK ou une CLI](#)
- [Utilisation AttachInternetGateway avec un AWS SDK ou une CLI](#)
- [Utilisation AttachNetworkInterface avec un AWS SDK ou une CLI](#)
- [Utilisation AttachVolume avec un AWS SDK ou une CLI](#)
- [Utilisation AttachVpnGateway avec un AWS SDK ou une CLI](#)
- [Utilisation AuthorizeSecurityGroupEgress avec un AWS SDK ou une CLI](#)
- [Utilisation AuthorizeSecurityGroupIngress avec un AWS SDK ou une CLI](#)
- [Utilisation CancelCapacityReservation avec un AWS SDK ou une CLI](#)
- [Utilisation CancelImportTask avec un AWS SDK ou une CLI](#)
- [Utilisation CancelSpotFleetRequests avec un AWS SDK ou une CLI](#)
- [Utilisation CancelSpotInstanceRequests avec un AWS SDK ou une CLI](#)
- [Utilisation ConfirmProductInstance avec un AWS SDK ou une CLI](#)
- [Utilisation CopyImage avec un AWS SDK ou une CLI](#)
- [Utilisation CopySnapshot avec un AWS SDK ou une CLI](#)

- [Utilisation CreateCapacityReservation avec un AWS SDK ou une CLI](#)
- [Utilisation CreateCustomerGateway avec un AWS SDK ou une CLI](#)
- [Utilisation CreateDhcpOptions avec un AWS SDK ou une CLI](#)
- [Utilisation CreateFlowLogs avec un AWS SDK ou une CLI](#)
- [Utilisation CreateImage avec un AWS SDK ou une CLI](#)
- [Utilisation CreateInstanceExportTask avec un AWS SDK ou une CLI](#)
- [Utilisation CreateInternetGateway avec un AWS SDK ou une CLI](#)
- [Utilisation CreateKeyPair avec un AWS SDK ou une CLI](#)
- [Utilisation CreateLaunchTemplate avec un AWS SDK ou une CLI](#)
- [Utilisation CreateNetworkAcl avec un AWS SDK ou une CLI](#)
- [Utilisation CreateNetworkAclEntry avec un AWS SDK ou une CLI](#)
- [Utilisation CreateNetworkInterface avec un AWS SDK ou une CLI](#)
- [Utilisation CreatePlacementGroup avec un AWS SDK ou une CLI](#)
- [Utilisation CreateRoute avec un AWS SDK ou une CLI](#)
- [Utilisation CreateRouteTable avec un AWS SDK ou une CLI](#)
- [Utilisation CreateSecurityGroup avec un AWS SDK ou une CLI](#)
- [Utilisation CreateSnapshot avec un AWS SDK ou une CLI](#)
- [Utilisation CreateSpotDatafeedSubscription avec un AWS SDK ou une CLI](#)
- [Utilisation CreateSubnet avec un AWS SDK ou une CLI](#)
- [Utilisation CreateTags avec un AWS SDK ou une CLI](#)
- [Utilisation CreateVolume avec un AWS SDK ou une CLI](#)
- [Utilisation CreateVpc avec un AWS SDK ou une CLI](#)
- [Utilisation CreateVpcEndpoint avec un AWS SDK ou une CLI](#)
- [Utilisation CreateVpnConnection avec un AWS SDK ou une CLI](#)
- [Utilisation CreateVpnConnectionRoute avec un AWS SDK ou une CLI](#)
- [Utilisation CreateVpnGateway avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteCustomerGateway avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteDhcpOptions avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteFlowLogs avec un AWS SDK ou une CLI](#)

- [Utilisation DeleteInternetGateway avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteKeyPair avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteLaunchTemplate avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteNetworkAcl avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteNetworkAclEntry avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteNetworkInterface avec un AWS SDK ou une CLI](#)
- [Utilisation DeletePlacementGroup avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteRoute avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteRouteTable avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteSecurityGroup avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteSnapshot avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteSpotDatafeedSubscription avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteSubnet avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteTags avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteVolume avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteVpc avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteVpnConnection avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteVpnConnectionRoute avec un AWS SDK ou une CLI](#)
- [Utilisation DeleteVpnGateway avec un AWS SDK ou une CLI](#)
- [Utilisation DeregisterImage avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeAccountAttributes avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeAddresses avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeAvailabilityZones avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeBundleTasks avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeCapacityReservations avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeCustomerGateways avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeDhcpOptions avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeFlowLogs avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeHostReservationOfferings avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeHosts avec un AWS SDK ou une CLI](#)

- [Utilisation DescribeInstanceProfileAssociations avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeIdFormat avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeIdentityIdFormat avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeImageAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeImages avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeImportImageTasks avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeImportSnapshotTasks avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeInstanceAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeInstanceStatus avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeInstanceTypes avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeInstances avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeInternetGateways avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeKeyPairs avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeNetworkAcls avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeNetworkInterfaceAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeNetworkInterfaces avec un AWS SDK ou une CLI](#)
- [Utilisation DescribePlacementGroups avec un AWS SDK ou une CLI](#)
- [Utilisation DescribePrefixLists avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeRegions avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeRouteTables avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeScheduledInstanceAvailability avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeScheduledInstances avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeSecurityGroups avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeSnapshotAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeSnapshots avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeSpotDatafeedSubscription avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeSpotFleetInstances avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeSpotFleetRequestHistory avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeSpotFleetRequests avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeSpotInstanceRequests avec un AWS SDK ou une CLI](#)

- [Utilisation DescribeSpotPriceHistory avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeSubnets avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeTags avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeVolumeAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeVolumeStatus avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeVolumes avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeVpcAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeVpcClassicLink avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeVpcClassicLinkDnsSupport avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeVpcEndpointServices avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeVpcEndpoints avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeVpcs avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeVpnConnections avec un AWS SDK ou une CLI](#)
- [Utilisation DescribeVpnGateways avec un AWS SDK ou une CLI](#)
- [Utilisation DetachInternetGateway avec un AWS SDK ou une CLI](#)
- [Utilisation DetachNetworkInterface avec un AWS SDK ou une CLI](#)
- [Utilisation DetachVolume avec un AWS SDK ou une CLI](#)
- [Utilisation DetachVpnGateway avec un AWS SDK ou une CLI](#)
- [Utilisation DisableVgwRoutePropagation avec un AWS SDK ou une CLI](#)
- [Utilisation DisableVpcClassicLink avec un AWS SDK ou une CLI](#)
- [Utilisation DisableVpcClassicLinkDnsSupport avec un AWS SDK ou une CLI](#)
- [Utilisation DisassociateAddress avec un AWS SDK ou une CLI](#)
- [Utilisation DisassociateRouteTable avec un AWS SDK ou une CLI](#)
- [Utilisation EnableVgwRoutePropagation avec un AWS SDK ou une CLI](#)
- [Utilisation EnableVolumelo avec un AWS SDK ou une CLI](#)
- [Utilisation EnableVpcClassicLink avec un AWS SDK ou une CLI](#)
- [Utilisation EnableVpcClassicLinkDnsSupport avec un AWS SDK ou une CLI](#)
- [Utilisation GetConsoleOutput avec un AWS SDK ou une CLI](#)
- [Utilisation GetHostReservationPurchasePreview avec un AWS SDK ou une CLI](#)
- [Utilisation GetPasswordData avec un AWS SDK ou une CLI](#)

- [Utilisation ImportImage avec un AWS SDK ou une CLI](#)
- [Utilisation ImportKeyPair avec un AWS SDK ou une CLI](#)
- [Utilisation ImportSnapshot avec un AWS SDK ou une CLI](#)
- [Utilisation ModifyCapacityReservation avec un AWS SDK ou une CLI](#)
- [Utilisation ModifyHosts avec un AWS SDK ou une CLI](#)
- [Utilisation ModifyIdFormat avec un AWS SDK ou une CLI](#)
- [Utilisation ModifyImageAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation ModifyInstanceAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation ModifyInstanceCreditSpecification avec un AWS SDK ou une CLI](#)
- [Utilisation ModifyNetworkInterfaceAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation ModifyReservedInstances avec un AWS SDK ou une CLI](#)
- [Utilisation ModifySnapshotAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation ModifySpotFleetRequest avec un AWS SDK ou une CLI](#)
- [Utilisation ModifySubnetAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation ModifyVolumeAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation ModifyVpcAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation MonitorInstances avec un AWS SDK ou une CLI](#)
- [Utilisation MoveAddressToVpc avec un AWS SDK ou une CLI](#)
- [Utilisation PurchaseHostReservation avec un AWS SDK ou une CLI](#)
- [Utilisation PurchaseScheduledInstances avec un AWS SDK ou une CLI](#)
- [Utilisation RebootInstances avec un AWS SDK ou une CLI](#)
- [Utilisation RegisterImage avec un AWS SDK ou une CLI](#)
- [Utilisation RejectVpcPeeringConnection avec un AWS SDK ou une CLI](#)
- [Utilisation ReleaseAddress avec un AWS SDK ou une CLI](#)
- [Utilisation ReleaseHosts avec un AWS SDK ou une CLI](#)
- [Utilisation ReplacelamInstanceProfileAssociation avec un AWS SDK ou une CLI](#)
- [Utilisation ReplaceNetworkAclAssociation avec un AWS SDK ou une CLI](#)
- [Utilisation ReplaceNetworkAclEntry avec un AWS SDK ou une CLI](#)
- [Utilisation ReplaceRoute avec un AWS SDK ou une CLI](#)
- [Utilisation ReplaceRouteTableAssociation avec un AWS SDK ou une CLI](#)

- [Utilisation ReportInstanceStatus avec un AWS SDK ou une CLI](#)
- [Utilisation RequestSpotFleet avec un AWS SDK ou une CLI](#)
- [Utilisation RequestSpotInstances avec un AWS SDK ou une CLI](#)
- [Utilisation ResetImageAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation ResetInstanceAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation ResetNetworkInterfaceAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation ResetSnapshotAttribute avec un AWS SDK ou une CLI](#)
- [Utilisation RevokeSecurityGroupEgress avec un AWS SDK ou une CLI](#)
- [Utilisation RevokeSecurityGroupIngress avec un AWS SDK ou une CLI](#)
- [Utilisation RunInstances avec un AWS SDK ou une CLI](#)
- [Utilisation RunScheduledInstances avec un AWS SDK ou une CLI](#)
- [Utilisation StartInstances avec un AWS SDK ou une CLI](#)
- [Utilisation StopInstances avec un AWS SDK ou une CLI](#)
- [Utilisation TerminateInstances avec un AWS SDK ou une CLI](#)
- [Utilisation UnassignPrivateIpAddresses avec un AWS SDK ou une CLI](#)
- [Utilisation UnmonitorInstances avec un AWS SDK ou une CLI](#)

Utilisation **AcceptVpcPeeringConnection** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `AcceptVpcPeeringConnection`.

CLI

AWS CLI

Pour accepter une connexion d'appairage VPC

Cet exemple accepte la demande de connexion d'appairage VPC spécifiée.

Commande :

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

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [AcceptVpcPeeringConnection](#) à la section Référence des AWS CLI commandes.

PowerShell**Outils pour PowerShell**

Exemple 1 : Cet exemple approuve le fichier VpcPeeringConnectionId pcx-1dfad234b56ff78be demandé

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

Sortie :

```
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
```

- Pour plus de détails sur l'API, reportez-vous [AcceptVpcPeeringConnection](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **AllocateAddress** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `AllocateAddress`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Démarrer avec les instances](#)

.NET

AWS SDK for .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

- Pour plus de détails sur l'API, voir [AllocateAddress](#) la section Référence des AWS SDK for .NET API.

Bash

AWS CLI avec le script Bash

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
#####
# 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
}

```

Fonctions utilitaires utilisées dans cet exemple.

```

#####
# 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
}
```

- Pour plus de détails sur l'API, reportez-vous [AllocateAddress](#) à la section Référence des AWS CLI commandes.

C++

SDK pour C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

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

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

- Pour plus de détails sur l'API, voir [AllocateAddress](#) la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Exemple 1 : allocation d'une adresse IP Elastic à partir d'un groupe d'adresses d'Amazon

L'exemple suivant `allocate-address` alloue une adresse IP Elastic. Amazon EC2 sélectionne l'adresse dans le pool d'adresses d'Amazon.

```
aws ec2 allocate-address
```

Sortie :

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

Pour plus d'informations, veuillez consulter la rubrique [Adresses IP Elastic](#) dans le Guide de l'utilisateur Amazon EC2.

Exemple 2 : allocation d'une adresse IP Elastic et l'associer à un groupe de frontières réseau

L'exemple `allocate-address` suivant alloue une adresse IP Elastic et l'associe au groupe de frontières réseau spécifié.

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

Sortie :

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

Pour plus d'informations, veuillez consulter la rubrique [Adresses IP Elastic](#) dans le Guide de l'utilisateur Amazon EC2.

Exemple 3 : allocation d'une adresse IP Elastic à partir de votre propre groupe d'adresses IP

L'exemple `allocate-address` suivant alloue une adresse IP Elastic à partir d'un groupe d'adresses IP que vous avez apporté à votre compte Amazon Web Services. Amazon EC2 sélectionne l'adresse dans le groupe d'adresses.

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

Sortie :

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

Pour plus d'informations, veuillez consulter la rubrique [Adresses IP Elastic](#) dans le Guide de l'utilisateur Amazon EC2.

- Pour plus de détails sur l'API, reportez-vous [AllocateAddress](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String allocateAddress(Ec2Client ec2) {  
  try {
```

```
        AllocateAddressRequest allocateRequest =
AllocateAddressRequest.builder()
        .domain(DomainType.VPC)
        .build();

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

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

- Pour plus de détails sur l'API, voir [AllocateAddress](#) la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

export const main = async () => {
    const command = new AllocateAddressCommand({});

    try {
        const { AllocationId, PublicIp } = await client.send(command);
        console.log("A new IP address has been allocated to your account:");
        console.log(`ID: ${AllocationId} Public IP: ${PublicIp}`);
    }
}
```

```
console.log(
  "You can view your IP addresses in the AWS Management Console for Amazon
  EC2. Look under Network & Security > Elastic IPs",
);
} catch (err) {
  console.error(err);
}
};
```

- Pour plus de détails sur l'API, voir [AllocateAddress](#) la section Référence des AWS SDK for JavaScript API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun getAllocateAddress(instanceIdVal: String?): String? {
    val allocateRequest =
        AllocateAddressRequest {
            domain = DomainType.Vpc
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val allocateResponse = ec2.allocateAddress(allocateRequest)
        val allocationIdVal = allocateResponse.allocationId

        val request =
            AssociateAddressRequest {
                instanceId = instanceIdVal
                allocationId = allocationIdVal
            }

        val associateResponse = ec2.associateAddress(request)
        return associateResponse.associationId
    }
}
```

```
    }
}
```

- Pour plus de détails sur l'API, consultez [AllocateAddress](#) la section AWS SDK pour la référence de l'API Kotlin.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple alloue une adresse IP élastique à utiliser avec une instance dans un VPC.

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

Sortie :

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

Exemple 2 : Cet exemple alloue une adresse IP élastique à utiliser avec une instance dans EC2-Classic.

```
New-EC2Address
```

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [AllocateAddress](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Python

SDK pour Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

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

    def allocate(self):
        """
        Allocates an Elastic IP address that can be associated with an Amazon EC2
        instance. By using an Elastic IP address, you can keep the public IP
        address
        constant even when you restart the associated instance.

        :return: The newly created Elastic IP object. By default, the address is
        not
```

```

        associated with any instance.
    """
    try:
        response =
self.ec2_resource.meta.client.allocate_address(Domain="vpc")
        self.elastic_ip =
self.ec2_resource.VpcAddress(response["AllocationId"])
    except ClientError as err:
        logger.error(
            "Couldn't allocate Elastic IP. Here's why: %s: %s",
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
    else:
        return self.elastic_ip

```

- Pour plus de détails sur l'API, consultez [AllocateAddress](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

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

```

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

- Pour plus de détails sur l'API, voir [AllocateAddress](#) la section Référence des AWS SDK for Ruby API.

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
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.
```

- Pour plus de détails sur l'API, reportez-vous [AllocateAddress](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **AllocateHosts** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `AllocateHosts`.

CLI

AWS CLI

Exemple 1 : pour allouer un hôte dédié

L'`allocate-hostsexemple` suivant alloue un seul hôte dédié dans la zone de `eu-west-1a` disponibilité, sur lequel vous pouvez lancer `m5.large` des instances. Par défaut, l'hôte dédié accepte uniquement les lancements d'instances cibles et ne prend pas en charge la restauration de l'hôte.

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

Sortie :

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

Exemple 2 : pour allouer un hôte dédié avec le placement automatique et la restauration de l'hôte activés

L'`allocate-hostsexemple` suivant alloue un seul hôte dédié dans la zone de `eu-west-1a` disponibilité avec le placement automatique et la restauration de l'hôte activés.

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

Sortie :

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

```
    "h-07879acf49EXAMPLE"  
  ]  
}
```

Exemple 3 : pour allouer un hôte dédié avec des balises

L'`allocate-hostsexemple` suivant alloue un seul hôte dédié et applique une balise avec une clé nommée `purpose` et une valeur de `production`

```
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}'
```

Sortie :

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

Pour plus d'informations, consultez la section [Allocation d'hôtes dédiés](#) dans le guide de l'utilisateur Amazon Elastic Compute Cloud pour les instances Linux.

- Pour plus de détails sur l'API, reportez-vous [AllocateHosts](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple alloue un hôte dédié à votre compte pour le type d'instance et la zone de disponibilité donnés

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

Sortie :

```
h-01e23f4cd567890f3
```

- Pour plus de détails sur l'API, reportez-vous [AllocateHosts](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **AssignPrivateIpAddresses** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `AssignPrivateIpAddresses`.

CLI

AWS CLI

Pour attribuer une interface réseau à une adresse IP privée secondaire spécifique

Cet exemple attribue l'adresse IP privée secondaire spécifiée à l'interface réseau spécifiée. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

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

Pour attribuer des adresses IP privées secondaires sélectionnées par Amazon EC2 à une interface réseau

Cet exemple attribue deux adresses IP privées secondaires à l'interface réseau spécifiée. Amazon EC2 attribue automatiquement ces adresses IP à partir des adresses IP disponibles dans la plage de blocs CIDR du sous-réseau auquel l'interface réseau est associée. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

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

- Pour plus de détails sur l'API, reportez-vous [AssignPrivateIpAddresses](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple attribue l'adresse IP privée secondaire spécifiée à l'interface réseau spécifiée.

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

Exemple 2 : Cet exemple crée deux adresses IP privées secondaires et les attribue à l'interface réseau spécifiée.

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

- Pour plus de détails sur l'API, reportez-vous [AssignPrivateIpAddresses](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **AssociateAddress** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `AssociateAddress`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Démarrer avec les instances](#)

.NET

AWS SDK for .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Associate an Elastic IP address to an EC2 instance.
/// </summary>
/// <param name="allocationId">The allocation Id of an Elastic IP address.</
param>
/// <param name="instanceId">The instance Id of the EC2 instance to
/// associate the address with.</param>
/// <returns>The association Id that represents
/// the association of the Elastic IP address with an instance.</returns>
public async Task<string> AssociateAddress(string allocationId, string
instanceId)
{
    var request = new AssociateAddressRequest
    {
        AllocationId = allocationId,
        InstanceId = instanceId
    };

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

- Pour plus de détails sur l'API, voir [AssociateAddress](#) la section Référence des AWS SDK for .NET API.

Bash

AWS CLI avec le script Bash

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

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

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

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

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

echo "$response"
return 0
}
```

Fonctions utilitaires utilisées dans cet exemple.

```
#####
# 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;
}
```

- Pour plus de détails sur l'API, reportez-vous [AssociateAddress](#) à la section Référence des AWS CLI commandes.

C++

SDK pour C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

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

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

- Pour plus de détails sur l'API, voir [AssociateAddress](#) la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Allocation d'une adresse IP Elastic dans EC2-Classic

Cet exemple associe une adresse IP Elastic à une instance dans EC2-Classic. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

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

Allocation d'une adresse IP Elastic dans EC2-VPC

Cet exemple associe une adresse IP Elastic à une instance dans un VPC.

Commande :

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

Sortie :

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

Cet exemple associe une adresse IP Elastic à une interface réseau.

Commande :

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

Cet exemple associe une adresse IP Elastic à une adresse IP privée, elle-même associée à une interface réseau.

Commande :

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

- Pour plus de détails sur l'API, reportez-vous [AssociateAddress](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

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

- Pour plus de détails sur l'API, voir [AssociateAddress](#) la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

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

  try {
    const { AssociationId } = await client.send(command);
    console.log(
      `Address with allocation ID ${allocationId} is now associated with instance
      ${instanceId}.`,
      `The association ID is ${AssociationId}.`,
    );
  } catch (err) {
    console.error(err);
  }
};
```

- Pour plus de détails sur l'API, voir [AssociateAddress](#) la section Référence des AWS SDK for JavaScript API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun associateAddressSc(
    instanceIdVal: String?,
    allocationIdVal: String?,
): String? {
    val associateRequest =
        AssociateAddressRequest {
            instanceId = instanceIdVal
            allocationId = allocationIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val associateResponse = ec2.associateAddress(associateRequest)
        return associateResponse.associationId
    }
}
```

- Pour plus de détails sur l'API, consultez [AssociateAddress](#) la section AWS SDK pour la référence de l'API Kotlin.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple associe l'adresse IP élastique spécifiée à l'instance spécifiée dans un VPC.

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

Sortie :

```
eipassoc-12345678
```

Exemple 2 : Cet exemple associe l'adresse IP élastique spécifiée à l'instance spécifiée dans EC2-Classic.

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

- Pour plus de détails sur l'API, reportez-vous [AssociateAddress](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Python

SDK pour Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

```
self.elastic_ip = elastic_ip

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

def associate(self, instance):
    """
    Associates an Elastic IP address with an instance. When this association
    is created, the Elastic IP's public IP address is immediately used as the
    public IP address of the associated instance.

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

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

- Pour plus de détails sur l'API, consultez [AssociateAddress](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

```
end
```

- Pour plus de détails sur l'API, voir [AssociateAddress](#) la section Référence des AWS SDK for Ruby API.

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
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.
```

- Pour plus de détails sur l'API, reportez-vous [AssociateAddress](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **AssociateDhcpOptions** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `AssociateDhcpOptions`.

CLI

AWS CLI

Pour associer un ensemble d'options DHCP à votre VPC

Cet exemple associe le jeu d'options DHCP spécifié au VPC spécifié. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

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

Pour associer les options DHCP définies par défaut à votre VPC

Cet exemple associe les options DHCP définies par défaut au VPC spécifié. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

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

- Pour plus de détails sur l'API, reportez-vous [AssociateDhcpOptions](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple associe le jeu d'options DHCP spécifié au VPC spécifié.

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

Exemple 2 : Cet exemple associe les options DHCP définies par défaut au VPC spécifié.

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

- Pour plus de détails sur l'API, reportez-vous [AssociateDhcpOptions](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **AssociateRouteTable** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `AssociateRouteTable`.

CLI

AWS CLI

Pour associer une table de routage à un sous-réseau

Cet exemple associe la table de routage spécifiée au sous-réseau spécifié.

Commande :

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

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [AssociateRouteTable](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple associe la table de routage spécifiée au sous-réseau spécifié.

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

Sortie :

```
rtbassoc-12345678
```

- Pour plus de détails sur l'API, reportez-vous [AssociateRouteTable](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **AttachInternetGateway** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `AttachInternetGateway`.

CLI

AWS CLI

Pour associer une passerelle Internet à votre VPC

L'`attach-internet-gateway` exemple suivant attache la passerelle Internet spécifiée au VPC spécifique.

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

Cette commande ne produit aucun résultat.

Pour plus d'informations, consultez [Passerelles Internet](#) dans le Guide de l'utilisateur Amazon VPC.

- Pour plus de détails sur l'API, reportez-vous [AttachInternetGateway](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple attache la passerelle Internet spécifiée au VPC spécifié.

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

Exemple 2 : Cet exemple crée un VPC et une passerelle Internet, puis attache la passerelle Internet au VPC.

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

- Pour plus de détails sur l'API, reportez-vous [AttachInternetGateway](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **AttachNetworkInterface** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `AttachNetworkInterface`.

CLI

AWS CLI

Exemple 1 : pour associer une interface réseau à une instance

L'`attach-network-interface` exemple suivant attache l'interface réseau spécifiée à l'instance spécifiée.

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

Sortie :

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

Pour plus d'informations, consultez la section [Elastic network interfaces](#) dans le guide de l'utilisateur Amazon EC2.

Exemple 2 : pour associer une interface réseau à une instance comportant plusieurs cartes réseau

L'`attach-network-interface` suivant attache l'interface réseau spécifiée à l'instance et à la carte réseau spécifiées.

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

Sortie :

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

Pour plus d'informations, consultez la section [Elastic network interfaces](#) dans le guide de l'utilisateur Amazon EC2.

- Pour plus de détails sur l'API, reportez-vous [AttachNetworkInterface](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple attache l'interface réseau spécifiée à l'instance spécifiée.

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

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [AttachNetworkInterface](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **AttachVolume** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `AttachVolume`.

CLI

AWS CLI

Pour attacher un volume à une instance

Cet exemple de commande attache un volume (`vol-1234567890abcdef0`) à une instance (`i-01474ef662b89480`) en tant que `/dev/sdf`.

Commande :

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

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [AttachVolume](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple attache le volume spécifié à l'instance spécifiée et l'expose avec le nom de périphérique spécifié.

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

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [AttachVolume](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **AttachVpnGateway** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `AttachVpnGateway`.

CLI

AWS CLI

Pour associer une passerelle privée virtuelle à votre VPC

L'`attach-vpn-gateway` exemple suivant attache la passerelle privée virtuelle spécifiée au VPC spécifié.

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

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [AttachVpnGateway](#) à la section Référence des AWS CLI commandes.

PowerShell**Outils pour PowerShell**

Exemple 1 : Cet exemple attache la passerelle privée virtuelle spécifiée au VPC spécifié.

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

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [AttachVpnGateway](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **AuthorizeSecurityGroupEgress** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `AuthorizeSecurityGroupEgress`.

CLI

AWS CLI

Pour ajouter une règle autorisant le trafic sortant vers une plage d'adresses spécifique

Cet exemple de commande ajoute une règle qui autorise l'accès aux plages d'adresses spécifiées sur le port TCP 80.

Commande (Linux) :

```
aws ec2 authorize-security-group-egress --group-id sg-1a2b3c4d --ip-permissions IpProtocol=tcp,FromPort=80,ToPort=80,IpRanges='[{"CidrIp=10.0.0.0/16}]'
```

Commande (Windows) :

```
aws ec2 authorize-security-group-egress --group-id sg-1a2b3c4d --ip-permissions IpProtocol=tcp,FromPort=80,ToPort=80,IpRanges=[{"CidrIp=10.0.0.0/16}]
```

Pour ajouter une règle autorisant le trafic sortant vers un groupe de sécurité spécifique

Cet exemple de commande ajoute une règle qui accorde l'accès au groupe de sécurité spécifié sur le port TCP 80.

Commande (Linux) :

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

Commande (Windows) :

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

- Pour plus de détails sur l'API, reportez-vous [AuthorizeSecurityGroupEgress](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple définit une règle de sortie pour le groupe de sécurité spécifié pour EC2-VPC. La règle autorise l'accès à la plage d'adresses IP spécifiée sur le port TCP 80. La syntaxe utilisée dans cet exemple nécessite PowerShell la version 3 ou supérieure.

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

Exemple 2 : avec PowerShell la version 2, vous devez utiliser New-Object pour créer l'IpPermission objet.

```
$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
```

Exemple 3 : Cet exemple accorde l'accès au groupe de sécurité source spécifié sur le port TCP 80.

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair  
$ug.GroupId = "sg-1a2b3c4d"  
$ug.UserId = "123456789012"  
  
Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission  
@( @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; UserIdGroupPairs=$ug } )
```

- Pour plus de détails sur l'API, reportez-vous [AuthorizeSecurityGroupEgress](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation `AuthorizeSecurityGroupIngress` avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `AuthorizeSecurityGroupIngress`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Démarrer avec les instances](#)

.NET

AWS SDK for .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Authorize the local computer ingress to EC2 instances associated
/// with the virtual private cloud (VPC) security group.
/// </summary>
/// <param name="groupName">The name of the security group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> AuthorizeSecurityGroupIngress(string groupName)
{
    // Get the IP address for the local computer.
    var ipAddress = await GetIpAddress();
    Console.WriteLine($"Your IP address is: {ipAddress}");
    var ipRanges = new List<IpRange> { new IpRange { CidrIp =
    $"{ipAddress}/32" } };
    var permission = new IpPermission
    {
        Ipv4Ranges = ipRanges,
        IpProtocol = "tcp",
        FromPort = 22,
        ToPort = 22
    }
}
```

```

    };
    var permissions = new List<IpPermission> { permission };
    var response = await _amazonEC2.AuthorizeSecurityGroupIngressAsync(
        new AuthorizeSecurityGroupIngressRequest(groupName, permissions));
    return response.HttpStatusCode == HttpStatusCode.OK;
}

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

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

```

- Pour plus de détails sur l'API, voir [AuthorizeSecurityGroupIngress](#) la section Référence des AWS SDK for .NET API.

Bash

AWS CLI avec le script Bash

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

#####
# 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
}

```

Fonctions utilitaires utilisées dans cet exemple.

```

#####
# 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
}
```

- Pour plus de détails sur l'API, reportez-vous [AuthorizeSecurityGroupIngress](#) à la section Référence des AWS CLI commandes.

C++

SDK pour C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

Aws::EC2::Model::IpPermission permission1;
permission1.SetIpProtocol("tcp");
permission1.SetToPort(80);
permission1.SetFromPort(80);
permission1.AddIpRanges(ip_range);

authorize_request.AddIpPermissions(permission1);

Aws::EC2::Model::IpPermission permission2;
```

```
permission2.SetIpProtocol("tcp");
permission2.SetToPort(22);
permission2.SetFromPort(22);
permission2.AddIpRanges(ip_range);

authorize_request.AddIpPermissions(permission2);

const Aws::EC2::Model::AuthorizeSecurityGroupIngressOutcome authorizeOutcome
=
    ec2Client.AuthorizeSecurityGroupIngress(authorizeRequest);

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

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

- Pour plus de détails sur l'API, voir [AuthorizeSecurityGroupIngress](#) la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Exemple 1 : ajout d'une règle autorisant le trafic SSH entrant

L'exemple `authorize-security-group-ingress` suivant ajoute une règle qui autorise le trafic entrant sur le port TCP 22 (SSH).

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

Sortie :

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

Exemple 2 : ajout d'une règle autorisant le trafic HTTP entrant en provenance d'un autre groupe de sécurité

L'exemple `authorize-security-group-ingress` suivant ajoute une règle qui autorise l'accès entrant sur le port TCP 80 à partir du groupe de sécurité source `sg-1a2b3c4d`. Le groupe source doit se trouver dans le même VPC ou dans un VPC homologue (nécessite une connexion d'appariement de VPC). Le trafic entrant est autorisé en fonction des adresses IP privées des instances associées au groupe de sécurité source (et non de l'adresse IP publique ou de l'adresse IP Elastic).

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

Sortie :

```
{
  "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"
        }
    }
]
}

```

Exemple 3 : ajout de plusieurs règles dans le même appel

L'exemple `authorize-security-group-ingress` suivant utilise le paramètre `ip-permissions` pour ajouter deux règles entrantes, l'une qui active l'accès entrant sur le port TCP 3389 (RDP) et l'autre qui active le ping/ICMP.

```

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

```

Sortie :

```

{
  "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"
    }
]
}

```

Exemple 4 : ajout d'une règle pour le trafic ICMP

L'exemple `authorize-security-group-ingress` suivant utilise le paramètre `ip-permissions` pour ajouter une règle entrante qui autorise le message ICMP Destination Unreachable: Fragmentation Needed and Don't Fragment was Set (type 3, code 4) depuis n'importe où.

```
aws ec2 authorize-security-group-ingress --group-id sg-1234567890abcdef0 --ip-permissions
=icmp, =3, =4, = "[{=0.0.0.0/0}] IpProtocol » FromPort ToPort IpRanges CidrIp
```

Sortie :

```

{
  "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"
    }
  ]
}

```

Exemple 5 : ajout d'une règle pour le trafic IPv6

L'exemple `authorize-security-group-ingress` suivant utilise le paramètre `ip-permissions` pour ajouter une règle entrante qui autorise l'accès SSH (port 22) à partir de la plage IPv6 `2001:db8:1234:1a00::/64`.

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

Sortie :

```
{
  "Return": true,
  "SecurityGroupRules": [
    {
      "SecurityGroupId": "sgr-0455bc68b60805563",
      "GroupId": "sg-1234567890abcdef0",
      "GroupOwnerId": "123456789012",
      "IsEgress": false,
      "IpProtocol": "tcp",
      "FromPort": 22,
      "ToPort": 22,
      "CidrIpv6": "2001:db8:1234:1a00::/64"
    }
  ]
}
```

Exemple 6 : ajout d'une règle pour le trafic ICMPv6

L'exemple `authorize-security-group-ingress` suivant utilise le paramètre `ip-permissions` pour ajouter une règle entrante qui autorise le trafic ICMPv6 depuis n'importe où.

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

Sortie :

```
{
  "Return": true,
  "SecurityGroupRules": [
    {
      "SecurityGroupId": "sgr-04b612d9363ab6327",
      "GroupId": "sg-1234567890abcdef0",
      "GroupOwnerId": "123456789012",
      "IsEgress": false,
      "IpProtocol": "icmpv6",
    }
  ]
}
```

```

        "FromPort": -1,
        "ToPort": -1,
        "CidrIpv6": ":::/0"
    }
]
}

```

Exemple 7 : ajout d'une règle avec une description

L'exemple `authorize-security-group-ingress` suivant utilise le paramètre `ip-permissions` pour ajouter une règle entrante qui autorise le trafic RDP depuis la plage d'adresses IPv4 spécifiée. La règle comprend une description qui vous aidera à l'identifier ultérieurement.

```
aws ec2 authorize-security-group-ingress --group-id sg-1234567890abcdef0 --ip-permissions IpProtocol =tcp, =3389, =3389, = "[{=203.0.113.0/24, FromPort description='Accès RDP depuis le bureau de New York'}] » ToPort IpRanges CidrIp
```

Sortie :

```

{
  "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"
    }
  ]
}

```

Exemple 8 : ajout d'une règle entrante utilisant une liste de préfixes

L'exemple `authorize-security-group-ingress` suivant utilise le paramètre `ip-permissions` pour ajouter une règle entrante qui autorise tout le trafic pour les plages CIDR de la liste de préfixes spécifiée.

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

Sortie :

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

Pour de plus amples informations, veuillez consulter [Groupes de sécurité](#) dans le Guide de l'utilisateur Amazon VPC (français non garanti).

- Pour plus de détails sur l'API, reportez-vous [AuthorizeSecurityGroupIngress](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String createSecurityGroup(Ec2Client ec2, String groupName,
String groupDesc, String vpcId,
    String myIpAddress) {
    try {
```

```
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

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

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

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

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

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

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

- Pour plus de détails sur l'API, voir [AuthorizeSecurityGroupIngress](#) la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

// Grant permissions for a single IP address to ssh into instances
// within the provided security group.
export const main = async () => {
  const command = new AuthorizeSecurityGroupIngressCommand({
    // Replace with a security group ID from the AWS console or
    // the DescribeSecurityGroupsCommand.
    GroupId: "SECURITY_GROUP_ID",
    IpPermissions: [
      {
        IpProtocol: "tcp",
        FromPort: 22,
        ToPort: 22,
        // Replace 0.0.0.0 with the IP address to authorize.
        // For more information on this notation, see
        // https://en.wikipedia.org/wiki/Classless_Inter-
        Domain_Routing#CIDR_notation
        IpRanges: [{ CidrIp: "0.0.0.0/32" }],
      },
    ],
  });

  try {
```

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

- Pour plus de détails sur l'API, voir [AuthorizeSecurityGroupIngress](#) la section Référence des AWS SDK for JavaScript API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createEC2SecurityGroupSc(
  groupNameVal: String?,
  groupDescVal: String?,
  vpcIdVal: String?,
  myIpAddress: String?,
): String? {
  val request =
    CreateSecurityGroupRequest {
      groupName = groupNameVal
      description = groupDescVal
      vpcId = vpcIdVal
    }

  Ec2Client { region = "us-west-2" }.use { ec2 ->
    val resp = ec2.createSecurityGroup(request)
    val ipRange =
      IpRange {
        cidrIp = "$myIpAddress/0"
      }
  }
```

```
    val ipPerm =
        IpPermission {
            ipProtocol = "tcp"
            toPort = 80
            fromPort = 80
            ipRanges = listOf(ipRange)
        }

    val ipPerm2 =
        IpPermission {
            ipProtocol = "tcp"
            toPort = 22
            fromPort = 22
            ipRanges = listOf(ipRange)
        }

    val authRequest =
        AuthorizeSecurityGroupIngressRequest {
            groupName = groupNameVal
            ipPermissions = listOf(ipPerm, ipPerm2)
        }
    ec2.authorizeSecurityGroupIngress(authRequest)
    println("Successfully added ingress policy to Security Group
    $groupNameVal")
    return resp.groupId
}
}
```

- Pour plus de détails sur l'API, consultez [AuthorizeSecurityGroupIngress](#) la section AWS SDK pour la référence de l'API Kotlin.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple définit les règles d'entrée pour un groupe de sécurité pour EC2-VPC. Ces règles accordent l'accès à une adresse IP spécifique pour le SSH (port 22) et le RDC (port 3389). Notez que vous devez identifier les groupes de sécurité pour EC2-VPC à l'aide de l'ID du groupe de sécurité et non du nom du groupe de sécurité. La syntaxe utilisée dans cet exemple nécessite PowerShell la version 3 ou supérieure.

```
$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 )
```

Exemple 2 : Avec PowerShell la version 2, vous devez utiliser `New-Object` pour créer les `IpPermission` objets.

```
$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 )
```

Exemple 3 : Cet exemple définit les règles d'entrée pour un groupe de sécurité pour EC2-Classic. Ces règles accordent l'accès à une adresse IP spécifique pour le SSH (port 22) et le RDC (port 3389). La syntaxe utilisée dans cet exemple nécessite PowerShell la version 3 ou supérieure.

```
$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 )
```

Exemple 4 : Avec PowerShell la version 2, vous devez utiliser `New-Object` pour créer les `IpPermission` objets.

```
$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 )

```

Exemple 5 : Cet exemple accorde l'accès au port TCP 8081 du groupe de sécurité source spécifié (sg-1a2b3c4d) au groupe de sécurité spécifié (sg-12345678).

```

$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"

Grant-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission
@( @{ IpProtocol="tcp"; FromPort="8081"; ToPort="8081"; UserIdGroupPairs=$ug } )

```

Exemple 6 : Cet exemple ajoute le CIDR 5.5.5.5/32 aux règles d'entrée du groupe de sécurité sg-1234abcd pour le trafic du port TCP 22 avec une description.

```

$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

```

- Pour plus de détails sur l'API, reportez-vous [AuthorizeSecurityGroupIngress](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Python

SDK pour Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

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

    def authorize_ingress(self, ssh_ingress_ip):
        """
        Adds a rule to the security group to allow access to SSH.

        :param ssh_ingress_ip: The IP address that is granted inbound access to
        connect
                               to port 22 over TCP, used for SSH.
        :return: The response to the authorization request. The 'Return' field of
        the
```

```
        response indicates whether the request succeeded or failed.
    """
    if self.security_group is None:
        logger.info("No security group to update.")
        return

    try:
        ip_permissions = [
            {
                # SSH ingress open to only the specified IP address.
                "IpProtocol": "tcp",
                "FromPort": 22,
                "ToPort": 22,
                "IpRanges": [{"CidrIp": f"{ssh_ingress_ip}/32"}],
            }
        ]
        response = self.security_group.authorize_ingress(
            IpPermissions=ip_permissions
        )
    except ClientError as err:
        logger.error(
            "Couldn't authorize inbound rules for %s. Here's why: %s: %s",
            self.security_group.id,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
    else:
        return response
```

- Pour plus de détails sur l'API, consultez [AuthorizeSecurityGroupIngress](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **CancelCapacityReservation** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CancelCapacityReservation`.

CLI

AWS CLI

Pour annuler une réservation de capacité

L'`cancel-capacity-reservation` suivant annule la réservation de capacité spécifiée.

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

Sortie :

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

Pour plus d'informations, consultez la section [Annulation d'une réservation de capacité](#) dans le guide de l'utilisateur Amazon Elastic Compute Cloud pour les instances Linux.

- Pour plus de détails sur l'API, reportez-vous [CancelCapacityReservation](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple annule la réservation de capacité `cr-0c1f2345db6f7cdba`

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

Sortie :

```
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
```

- Pour plus de détails sur l'API, reportez-vous [CancelCapacityReservation](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **CancelImportTask** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CancelImportTask`.

CLI

AWS CLI

Pour annuler une tâche d'importation

L'`cancel-import-task` exemple suivant annule la tâche d'importation d'image spécifiée.

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

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [CancelImportTask](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple annule la tâche d'importation spécifiée (capture d'écran ou importation d'image). Si nécessaire, une raison peut être fournie à l'aide du **-CancelReason** paramètre.

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

- Pour plus de détails sur l'API, reportez-vous [CancelImportTask](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation `CancelSpotFleetRequests` avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CancelSpotFleetRequests`.

CLI

AWS CLI

Exemple 1 : pour annuler une demande de flotte Spot et mettre fin aux instances associées

L'`cancel-spot-fleet-request` exemple suivant annule une demande de parc Spot et met fin aux instances à la demande et aux instances ponctuelles associées.

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

Sortie :

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

Pour plus d'informations, consultez [Annuler une demande de parc de spots](#) dans le guide de l'utilisateur Amazon Elastic Compute Cloud pour les instances Linux.

Exemple 2 : Pour annuler une demande de flotte Spot sans mettre fin aux instances associées

L'`cancel-spot-fleet-requests` exemple suivant annule une demande de parc Spot sans mettre fin aux instances à la demande et aux instances ponctuelles associées.

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

Sortie :

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

Pour plus d'informations, consultez [Annuler une demande de parc de spots](#) dans le guide de l'utilisateur Amazon Elastic Compute Cloud pour les instances Linux.

- Pour plus de détails sur l'API, reportez-vous [CancelSpotFleetRequests](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple annule la demande de parc Spot spécifiée et met fin aux instances Spot associées.

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

Exemple 2 : Cet exemple annule la demande de parc Spot spécifiée sans mettre fin aux instances Spot associées.

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

- Pour plus de détails sur l'API, reportez-vous [CancelSpotFleetRequests](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation `CancelSpotInstanceRequests` avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CancelSpotInstanceRequests`.

CLI

AWS CLI

Pour annuler les demandes d'instance Spot

Cet exemple de commande annule une demande d'instance Spot.

Commande :

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

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [CancelSpotInstanceRequests](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple annule la demande d'instance Spot spécifiée.

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

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [CancelSpotInstanceRequests](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **ConfirmProductInstance** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ConfirmProductInstance`.

CLI

AWS CLI

Pour confirmer l'instance du produit

Cet exemple détermine si le code produit spécifié est associé à l'instance spécifiée.

Commande :

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

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [ConfirmProductInstance](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple détermine si le code produit spécifié est associé à l'instance spécifiée.

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

- Pour plus de détails sur l'API, reportez-vous [ConfirmProductInstance](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **CopyImage** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CopyImage`.

CLI

AWS CLI

Exemple 1 : pour copier une AMI dans une autre région

L'`copy-image` exemple de commande suivant copie l'AMI spécifiée de la `us-west-2` région vers la `us-east-1` région et ajoute une brève description.

```
aws ec2 copy-image \  
  --region us-east-1 \  
  --source-image-id ami-12345678
```

```
--name ami-name \  
--source-region us-west-2 \  
--source-image-id ami-066877671789bd71b \  
--description "This is my copied image."
```

Sortie :

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

Pour plus d'informations, consultez [Copier une AMI](#) dans le guide de l'utilisateur Amazon EC2.

Exemple 2 : pour copier une AMI dans une autre région et chiffrer l'instantané de sauvegarde

La `copy-image` commande suivante copie l'AMI spécifiée de la `us-west-2` région vers la région actuelle et chiffre le snapshot de sauvegarde à l'aide de la clé KMS spécifiée.

```
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
```

Sortie :

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

Pour plus d'informations, consultez [Copier une AMI](#) dans le guide de l'utilisateur Amazon EC2.

Exemple 3 : pour inclure vos balises d'AMI définies par l'utilisateur lors de la copie d'une AMI

La `copy-image` commande suivante utilise le `--copy-image-tags` paramètre pour copier les balises AMI définies par l'utilisateur lors de la copie de l'AMI.

```
aws ec2 copy-image \  
  --region us-east-1 \  
  --name ami-name \  
  --source-region us-west-2 \  
  --copy-image-tags
```

```
--source-image-id ami-066877671789bd71b \  
--description "This is my copied image."  
--copy-image-tags
```

Sortie :

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

Pour plus d'informations, consultez [Copier une AMI](#) dans le guide de l'utilisateur Amazon EC2.

- Pour plus de détails sur l'API, reportez-vous [CopyImage](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple copie l'AMI spécifiée dans la région « UE (Irlande) » vers la région « USA Ouest (Oregon) ». Si -Region n'est pas spécifiée, la région par défaut actuelle est utilisée comme région de destination.

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

Sortie :

```
ami-87654321
```

- Pour plus de détails sur l'API, reportez-vous [CopyImage](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **CopySnapshot** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CopySnapshot`.

CLI

AWS CLI

Exemple 1 : pour copier un instantané dans une autre région

L'copy-snapshot exemple de commande suivant copie le cliché spécifié de la us-west-2 région vers la us-east-1 région et ajoute une brève description.

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

Sortie :

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

Pour plus d'informations, consultez [Copier un instantané Amazon EBS](#) dans le guide de l'utilisateur Amazon EC2.

Exemple 2 : pour copier un instantané non chiffré et chiffrer le nouveau cliché

La copy-snapshot commande suivante copie le cliché non chiffré spécifié de la us-west-2 région vers la région actuelle et chiffre le nouveau cliché à l'aide de la clé KMS spécifiée.

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

Sortie :

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

Pour plus d'informations, consultez [Copier un instantané Amazon EBS](#) dans le guide de l'utilisateur Amazon EC2.

- Pour plus de détails sur l'API, reportez-vous [CopySnapshot](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple copie le cliché spécifié de la région UE (Irlande) vers la région USA Ouest (Oregon).

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

Exemple 2 : Si vous définissez une région par défaut et que vous omettez le paramètre Région, la région de destination par défaut est la région par défaut.

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

- Pour plus de détails sur l'API, reportez-vous [CopySnapshot](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **CreateCapacityReservation** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CreateCapacityReservation`.

CLI

AWS CLI

Exemple 1 : pour créer une réservation de capacité

L'`create-capacity-reservation`exemple suivant crée une réservation de capacité dans la zone de `eu-west-1` à disponibilité, dans laquelle vous pouvez lancer trois `t2.medium`

instances exécutant un système d'exploitation Linux/Unix. Par défaut, la réservation de capacité est créée avec des critères de correspondance des instances ouvertes et ne prend pas en charge le stockage éphémère, et elle reste active jusqu'à ce que vous l'annuliez manuellement.

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

Sortie :

```
{  
  "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"  
  }  
}
```

Exemple 2 : pour créer une réservation de capacité qui se termine automatiquement à une date/heure spécifiée

L'`create-capacity-reservation` exemple suivant crée une réservation de capacité dans la zone de eu-west-1a disponibilité, dans laquelle vous pouvez lancer trois m5.large instances exécutant un système d'exploitation Linux/Unix. Cette réservation de capacité prend fin automatiquement le 31/08/2019 à 23:59:59.

```
aws ec2 create-capacity-reservation \  
  --availability-zone eu-west-1a \  
  --instance-type m5.large
```

```
--instance-type m5.large \  
--instance-platform Linux/UNIX \  
--instance-count 3 \  
--end-date-type limited \  
--end-date 2019-08-31T23:59:59Z
```

Sortie :

```
{  
  "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"  
  }  
}
```

Exemple 3 : pour créer une réservation de capacité qui accepte uniquement les lancements d'instances ciblés

L'create-capacity-reservationexemple suivant crée une réservation de capacité qui accepte uniquement les lancements d'instances ciblés.

```
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
```

Sortie :

```
{
  "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"
  }
}
```

Pour plus d'informations, consultez la section [Création d'une réservation de capacité](#) dans le guide de l'utilisateur Amazon Elastic Compute Cloud pour les instances Linux.

- Pour plus de détails sur l'API, reportez-vous [CreateCapacityReservation](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple crée une nouvelle réservation de capacité avec les attributs spécifiés

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

Sortie :

```
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
```

- Pour plus de détails sur l'API, reportez-vous [CreateCapacityReservation](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation `CreateCustomerGateway` avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CreateCustomerGateway`.

CLI

AWS CLI

Pour créer une passerelle client

Cet exemple crée une passerelle client avec l'adresse IP spécifiée pour son interface externe.

Commande :

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

Sortie :

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

```
    "BgpAsn": "65534"  
  }  
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateCustomerGateway](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple crée la passerelle client spécifiée.

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

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [CreateCustomerGateway](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **CreateDhcpOptions** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CreateDhcpOptions`.

CLI

AWS CLI

Pour créer un ensemble d'options DHCP

L'`create-dhcp-optionsexemple` suivant crée un ensemble d'options DHCP qui spécifie le nom de domaine, les serveurs de noms de domaine et le type de nœud NetBIOS.

```
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"
```

Sortie :

```
{  
  "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"  
  }  
}
```

```
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateDhcpOptions](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple crée le jeu d'options DHCP spécifié. La syntaxe utilisée dans cet exemple nécessite PowerShell la version 3 ou ultérieure.

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

Sortie :

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

Exemple 2 : Avec PowerShell la version 2, vous devez utiliser New-Object pour créer chaque option DHCP.

```
$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)
```

Sortie :

DhcpConfigurations	DhcpOptionsId	Tags
--------------------	---------------	------

```
-----
{domain-name, domain-name-servers}   dopt-2a3b4c5d   {}
```

- Pour plus de détails sur l'API, reportez-vous [CreateDhcpOptions](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **CreateFlowLogs** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CreateFlowLogs`.

CLI

AWS CLI

Exemple 1 : pour créer un journal de flux

L'exemple suivant crée un journal de flux qui capture tout le trafic rejeté pour l'interface réseau spécifiée. Les journaux de flux sont transmis à un groupe de CloudWatch journaux dans Logs à l'aide des autorisations associées au rôle IAM spécifié.

```
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
```

Sortie :

```
{
  "ClientToken": "so0eNA2uSHUN1HI0S2cJ305GuIX1CezaRdGtexample",
  "FlowLogIds": [
    "fl-12345678901234567"
  ],
  "Unsuccessful": []
}
```

Pour plus d'informations, consultez la rubrique [Journaux de flux VPC](#) dans le Guide de l'utilisateur Amazon VPC.

Exemple 2 : pour créer un journal de flux avec un format personnalisé

L'`create-flow-log` exemple suivant crée un journal de flux qui capture tout le trafic pour le VPC spécifié et transmet les journaux de flux à un compartiment Amazon S3. Le paramètre `--log-format` spécifie un format personnalisé pour les enregistrements de journal de flux. Pour exécuter cette commande sous Windows, remplacez les guillemets simples (') par des guillemets doubles («).

```
aws ec2 create-flow-logs \  
  --resource-type VPC \  
  --resource-ids vpc-00112233344556677 \  
  --traffic-type ALL \  
  --log-destination-type s3 \  
  --log-destination arn:aws:s3:::flow-log-bucket/my-custom-flow-logs/ \  
  --log-format '${version} ${vpc-id} ${subnet-id} ${instance-id} ${srcaddr} \  
  ${dstaddr} ${srcport} ${dstport} ${protocol} ${tcp-flags} ${type} ${pkt-srcaddr} \  
  ${pkt-dstaddr}'
```

Pour plus d'informations, consultez la rubrique [Journaux de flux VPC](#) dans le Guide de l'utilisateur Amazon VPC.

Exemple 3 : pour créer un journal de flux avec un intervalle d'agrégation maximal d'une minute

L'`create-flow-log` exemple suivant crée un journal de flux qui capture tout le trafic pour le VPC spécifié et transmet les journaux de flux à un compartiment Amazon S3. Le `--max-aggregation-interval` paramètre spécifie un intervalle d'agrégation maximal de 60 secondes (1 minute).

```
aws ec2 create-flow-logs \  
  --resource-type VPC \  
  --resource-ids vpc-00112233344556677 \  
  --traffic-type ALL \  
  --log-destination-type s3 \  
  --log-destination arn:aws:s3:::flow-log-bucket/my-custom-flow-logs/ \  
  --max-aggregation-interval 60
```

Pour plus d'informations, consultez la rubrique [Journaux de flux VPC](#) dans le Guide de l'utilisateur Amazon VPC.

- Pour plus de détails sur l'API, reportez-vous [CreateFlowLogs](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple crée un journal de flux EC2 pour le sous-réseau subnet-1d234567 vers le cloud-watch-log nom « subnet1-log » pour tout le trafic « REJECT » en utilisant les autorisations du rôle « Admin »

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

Sortie :

```
ClientToken                                FlowLogIds                                Unsuccessful
-----
m1VN2cxP3iB4qo//VUK15EU6cF7gQL0xcqNefvjeTGw= {f1-012fc34eed5678c9d} {}
```

- Pour plus de détails sur l'API, reportez-vous [CreateFlowLogs](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **CreateImage** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CreateImage`.

CLI

AWS CLI

Exemple 1 : pour créer une AMI à partir d'une instance basée sur Amazon EBS

L'`create-image`exemple suivant crée une AMI à partir de l'instance spécifiée.

```
aws ec2 create-image \  
  --instance-id i-1234567890abcdef0 \  
  --name "My server" \  
  --description "An AMI for my server"
```

Sortie :

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

Pour plus d'informations sur la spécification d'un mappage de périphérique en mode bloc pour votre AMI, consultez la section [Spécification d'un mappage de périphérique en mode bloc pour une AMI](#) dans le guide de l'utilisateur Amazon EC2.

Exemple 2 : pour créer une AMI à partir d'une instance basée sur Amazon EBS sans redémarrer

L'`create-image`exemple suivant crée une AMI et définit le paramètre `--no-reboot` afin que l'instance ne soit pas redémarrée avant la création de l'image.

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

Sortie :

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

Pour plus d'informations sur la spécification d'un mappage de périphérique en mode bloc pour votre AMI, consultez la section [Spécification d'un mappage de périphérique en mode bloc pour une AMI](#) dans le guide de l'utilisateur Amazon EC2.

Exemple 3 : pour étiqueter une AMI et des instantanés lors de leur création

L'`create-image`exemple suivant crée une AMI et étiquette l'AMI et les instantanés avec la même balise. `cost-center=cc123`

```
aws ec2 create-image \  
  --instance-id i-1234567890abcdef0 \  
  --name "My server" \  
  --tag-specifications "ResourceType=image,Tags=[{Key=cost-  
center,Value=cc123}]" "ResourceType=snapshot,Tags=[{Key=cost-  
center,Value=cc123}]"
```

Sortie :

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

Pour plus d'informations sur le balisage de vos ressources lors de leur création, consultez la section [Ajouter des balises lors de la création des ressources](#) dans le guide de l'utilisateur Amazon EC2.

- Pour plus de détails sur l'API, reportez-vous [CreateImage](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple crée une AMI avec le nom et la description spécifiés, à partir de l'instance spécifiée. Amazon EC2 tente d'arrêter correctement l'instance avant de créer l'image, puis redémarre l'instance une fois l'opération terminée.

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

Exemple 2 : Cet exemple crée une AMI avec le nom et la description spécifiés, à partir de l'instance spécifiée. Amazon EC2 crée l'image sans arrêter ni redémarrer l'instance ; par conséquent, l'intégrité du système de fichiers sur l'image créée ne peut être garantie.

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

Exemple 3 : Cet exemple crée une AMI avec trois volumes. Le premier volume est basé sur un instantané Amazon EBS. Le deuxième volume est un volume Amazon EBS vide de 100

GiB. Le troisième volume est un volume de stockage d'instance. La syntaxe utilisée dans cet exemple nécessite PowerShell la version 3 ou supérieure.

```
$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"})
```

- Pour plus de détails sur l'API, reportez-vous [CreateImage](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **CreateInstanceExportTask** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CreateInstanceExportTask`.

CLI

AWS CLI

Pour exporter une instance

Cet exemple de commande crée une tâche pour exporter l'instance `i-1234567890abcdef0` vers le compartiment `myexportbucket` d'Amazon S3.

Commande :

```
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
```

Sortie :

```
{
```

```

    "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"
    }
  }
}

```

- Pour plus de détails sur l'API, reportez-vous [CreateInstanceExportTask](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple exporte une instance arrêtée **i-0800b00a00EXAMPLE**, sous forme de disque dur virtuel (VHD) vers le compartiment **testbucket-export-instances-2019** S3. L'environnement cible est **Microsoft**, et le paramètre de région est ajouté parce que l'instance se trouve dans la **us-east-1** région, alors que la AWS région par défaut de l'utilisateur n'est pas us-east-1. Pour obtenir le statut de la tâche d'exportation, copiez la **ExportTaskId** valeur à partir des résultats de cette commande, puis exécutez **Get-EC2ExportTask -ExportTaskId export_task_ID_from_results**.

```

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

```

Sortie :

```

Description          :
ExportTaskId         : export-i-077c73108aEXAMPLE

```

```
ExportToS3Task      : Amazon.EC2.Model.ExportToS3Task
InstanceExportDetails : Amazon.EC2.Model.InstanceExportDetails
State               : active
StatusMessage       :
```

- Pour plus de détails sur l'API, reportez-vous [CreateInstanceExportTask](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation `CreateInternetGateway` avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CreateInternetGateway`.

CLI

AWS CLI

Pour créer une passerelle Internet

L'exemple suivant crée une passerelle Internet avec la balise `my-igw`.

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

Sortie :

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

```

    }
  ]
}
}

```

Pour plus d'informations, consultez [Passerelles Internet](#) dans le Guide de l'utilisateur Amazon VPC.

- Pour plus de détails sur l'API, reportez-vous [CreateInternetGateway](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple crée une passerelle Internet.

```
New-EC2InternetGateway
```

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [CreateInternetGateway](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **CreateKeyPair** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CreateKeyPair`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Démarrer avec les instances](#)

.NET

AWS SDK for .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

    var response = await _amazonEC2.CreateKeyPairAsync(request);

    if (response.HttpStatusCode == HttpStatusCode.OK)
    {
        var kp = response.KeyPair;
        return kp;
    }
    else
    {
        Console.WriteLine("Could not create key pair.");
        return null;
    }
}

/// <summary>
/// Save KeyPair information to a temporary file.
/// </summary>
/// <param name="keyPair">The name of the key pair.</param>
```

```

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

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

    return pemFileName;
}

```

- Pour plus de détails sur l'API, voir [CreateKeyPair](#) la section Référence des AWS SDK for .NET API.

Bash

AWS CLI avec le script Bash

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

#####
# function ec2_create_keypair
#
# This function creates an Amazon Elastic Compute Cloud (Amazon EC2) ED25519 or
# 2048-bit RSA key pair
# and writes it to a file.
#
# Parameters:
#     -n key_pair_name - A key pair name.
#     -f file_path - File to store the key pair.
#
# And:

```

```
#      0 - If successful.
#      1 - If it fails.
#####
function ec2_create_keypair() {
    local key_pair_name file_path response
    local option OPTARG # Required to use getopt command in a function.

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

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

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

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

```

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
}

```

Fonctions utilitaires utilisées dans cet exemple.

```

#####
# 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  
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateKeyPair](#) à la section Référence des AWS CLI commandes.

C++

SDK pour C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);  
Aws::EC2::Model::CreateKeyPairRequest request;  
request.SetKeyName(keyPairName);
```

```
Aws::EC2::Model::CreateKeyPairOutcome outcome =
ec2Client.CreateKeyPair(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Failed to create key pair:" <<
        outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Successfully created key pair named " <<
        keyPairName << std::endl;
}
```

- Pour plus de détails sur l'API, voir [CreateKeyPair](#) la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Création d'une paire de clés

Cet exemple crée une paire de clés nommée `MyKeyPair`.

Commande :

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

La sortie est une version ASCII de la clé privée et de l'empreinte de la clé. Vous devez enregistrer la clé dans un fichier.

Pour plus d'informations, veuillez consulter la rubrique Using Key Pairs dans le Guide de l'utilisateur AWS Command Line Interface.

- Pour plus de détails sur l'API, reportez-vous [CreateKeyPair](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

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

- Pour plus de détails sur l'API, voir [CreateKeyPair](#) la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

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

- Pour plus de détails sur l'API, voir [CreateKeyPair](#) la section Référence des AWS SDK for JavaScript API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createEC2KeyPair(keyNameVal: String) {
    val request =
        CreateKeyPairRequest {
            keyName = keyNameVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.createKeyPair(request)
        println("The key ID is ${response.keyPairId}")
    }
}
```

- Pour plus de détails sur l'API, consultez [CreateKeyPair](#) la section AWS SDK pour la référence de l'API Kotlin.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple crée une paire de clés et capture la clé privée RSA codée PEM dans un fichier portant le nom spécifié. Lorsque vous l'utilisez PowerShell, le codage doit être défini sur ASCII pour générer une clé valide. Pour plus d'informations, consultez la section Créer, afficher et supprimer des paires de clés Amazon EC2 (<https://docs.aws.amazon.com/cli/latest/userguide/cli-services-ec2-keypairs.html>) dans le guide de l'utilisateur de l'interface de ligne de commande AWS.

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

- Pour plus de détails sur l'API, reportez-vous [CreateKeyPair](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Python

SDK pour Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

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

    def create(self, key_name):
```

```
"""
    Creates a key pair that can be used to securely connect to an EC2
instance.
    The returned key pair contains private key information that cannot be
retrieved
again. The private key data is stored as a .pem file.

:param key_name: The name of the key pair to create.
:return: A Boto3 KeyPair object that represents the newly created key
pair.
"""
try:
    self.key_pair = self.ec2_resource.create_key_pair(KeyName=key_name)
    self.key_file_path = os.path.join(
        self.key_file_dir.name, f"{self.key_pair.name}.pem"
    )
    with open(self.key_file_path, "w") as key_file:
        key_file.write(self.key_pair.key_material)
except ClientError as err:
    logger.error(
        "Couldn't create key %s. Here's why: %s: %s",
        key_name,
        err.response["Error"]["Code"],
        err.response["Error"]["Message"],
    )
    raise
else:
    return self.key_pair
```

- Pour plus de détails sur l'API, consultez [CreateKeyPair](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
# This code example does the following:
# 1. Creates a key pair in Amazon Elastic Compute Cloud (Amazon EC2).
# 2. Displays information about available key pairs.
# 3. Deletes the key pair.

require "aws-sdk-ec2"

# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param key_pair_name [String] The name for the key pair and private
#   key file.
# @return [Boolean] true if the key pair and private key file were
#   created; otherwise, false.
# @example
#   exit 1 unless key_pair_created?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-key-pair'
#   )
def key_pair_created?(ec2_client, key_pair_name)
  key_pair = ec2_client.create_key_pair(key_name: key_pair_name)
  puts "Created key pair '#{key_pair.key_name}' with fingerprint " \
    "'#{key_pair.key_fingerprint}' and ID '#{key_pair.key_pair_id}'."
  filename = File.join(Dir.home, key_pair_name + ".pem")
  File.open(filename, "w") { |file| file.write(key_pair.key_material) }
  puts "Private key file saved locally as '#{filename}'."
  return true
rescue Aws::EC2::Errors::InvalidKeyPairDuplicate
  puts "Error creating key pair: a key pair named '#{key_pair_name}' " \
    "already exists."
  return false
rescue StandardError => e
  puts "Error creating key pair or saving private key file: #{e.message}"
end
```

```
    return false
  end

  # Displays information about available key pairs in
  # Amazon Elastic Compute Cloud (Amazon EC2).
  #
  # @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
  # @example
  #   describe_key_pairs(Aws::EC2::Client.new(region: 'us-west-2'))
  def describe_key_pairs(ec2_client)
    result = ec2_client.describe_key_pairs
    if result.key_pairs.count.zero?
      puts "No key pairs found."
    else
      puts "Key pair names:"
      result.key_pairs.each do |key_pair|
        puts key_pair.key_name
      end
    end
  end
rescue StandardError => e
  puts "Error getting information about key pairs: #{e.message}"
end

# Deletes a key pair in Amazon Elastic Compute Cloud (Amazon EC2).
#
# Prerequisites:
#
# - The key pair to delete.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param key_pair_name [String] The name of the key pair to delete.
# @return [Boolean] true if the key pair was deleted; otherwise, false.
# @example
#   exit 1 unless key_pair_deleted?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-key-pair'
#   )
def key_pair_deleted?(ec2_client, key_pair_name)
  ec2_client.delete_key_pair(key_name: key_pair_name)
  return true
rescue StandardError => e
  puts "Error deleting key pair: #{e.message}"
  return false
end
```

```
# Example usage:
def run_me
  key_pair_name = ""
  region = ""
  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
    puts "Usage: ruby ec2-ruby-example-key-pairs.rb KEY_PAIR_NAME REGION"
    puts "Example: ruby ec2-ruby-example-key-pairs.rb my-key-pair us-west-2"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    key_pair_name = "my-key-pair"
    region = "us-west-2"
  # Otherwise, use the values as specified at the command prompt.
  else
    key_pair_name = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts "Displaying existing key pair names before creating this key pair..."
  describe_key_pairs(ec2_client)

  puts "-" * 10
  puts "Creating key pair..."
  unless key_pair_created?(ec2_client, key_pair_name)
    puts "Stopping program."
    exit 1
  end

  puts "-" * 10
  puts "Displaying existing key pair names after creating this key pair..."
  describe_key_pairs(ec2_client)

  puts "-" * 10
  puts "Deleting key pair..."
  unless key_pair_deleted?(ec2_client, key_pair_name)
    puts "Stopping program. You must delete the key pair yourself."
    exit 1
  end
  puts "Key pair deleted."
```

```
puts "-" * 10
puts "Now that the key pair is deleted, " \
     "also deleting the related private key pair file..."
filename = File.join(Dir.home, key_pair_name + ".pem")
File.delete(filename)
if File.exist?(filename)
  puts "Could not delete file at '#{filename}'. You must delete it yourself."
else
  puts "File deleted."
end

puts "-" * 10
puts "Displaying existing key pair names after deleting this key pair..."
describe_key_pairs(ec2_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Pour plus de détails sur l'API, voir [CreateKeyPair](#) la section Référence des AWS SDK for Ruby API.

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
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.
```

- Pour plus de détails sur l'API, reportez-vous [CreateKeyPair](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation `CreateLaunchTemplate` avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CreateLaunchTemplate`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Créer et gérer un service résilient](#)

.NET

AWS SDK for .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling.
/// The launch template specifies a Bash script in its user data field that
runs after
/// the instance is started. This script installs the Python packages and
starts a Python
/// web server on the instance.
/// </summary>
```

```
    /// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
    /// <param name="instancePolicyPath">The path to a permissions policy to
create and attach to the profile.</param>
    /// <returns>The template object.</returns>
    public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
    {
        await CreateKeyPair(_keyPairName);
        await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName, _instanceProfileName, instancePolicyPath);

        var startServerText = await File.ReadAllTextAsync(startupScriptPath);
        var plainTextBytes = System.Text.Encoding.UTF8.GetBytes(startServerText);

        var amiLatest = await _amazonSsm.GetParameterAsync(
            new GetParameterRequest() { Name = _amiParam });
        var amiId = amiLatest.Parameter.Value;
        var launchTemplateResponse = await _amazonEc2.CreateLaunchTemplateAsync(
            new CreateLaunchTemplateRequest()
            {
                LaunchTemplateName = _launchTemplateName,
                LaunchTemplateData = new RequestLaunchTemplateData()
                {
                    InstanceType = _instanceType,
                    ImageId = amiId,
                    IamInstanceProfile =
                        new
LaunchTemplateIamInstanceProfileSpecificationRequest()
                        {
                            Name = _instanceProfileName
                        },
                    KeyName = _keyPairName,
                    UserData = System.Convert.ToBase64String(plainTextBytes)
                }
            }
        ));
        return launchTemplateResponse.LaunchTemplate;
    }
}
```

- Pour plus de détails sur l'API, voir [CreateLaunchTemplate](#) la section Référence des AWS SDK for .NET API.

CLI

AWS CLI

Exemple 1 : création d'un modèle de lancement

L'exemple `create-launch-template` suivant crée un modèle de lancement qui spécifie le sous-réseau dans lequel l'instance doit être lancée, attribue une adresse IP publique et une adresse IPv6 à l'instance et crée une balise pour l'instance.

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

Sortie :

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

Pour de plus amples informations, veuillez consulter la rubrique Lancer une instance à partir d'un modèle de lancement dans le Guide de l'utilisateur Amazon Elastic Compute Cloud. Pour plus d'informations sur l'utilisation de guillemets avec les paramètres au format JSON, veuillez consulter la rubrique Quoting Strings dans le Guide de l'utilisateur AWS Command Line Interface.

Exemple 2 : création d'un modèle de lancement pour Amazon EC2 Auto Scaling

L'exemple `create-launch-template` suivant crée un modèle de lancement avec plusieurs balises et un mappage de périphérique de stockage en mode bloc pour spécifier un volume EBS supplémentaire lors du lancement d'une instance. Spécifiez une valeur pour `Groups` correspondant à des groupes de sécurité pour le VPC dans lequel votre groupe Auto Scaling lancera des instances. Spécifiez le VPC et les sous-réseaux en tant que propriétés du groupe Auto Scaling.

```
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
```

Sortie :

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

Pour plus d'informations, veuillez consulter la rubrique [Créer un modèle de lancement pour un groupe Auto Scaling](#) dans le Guide de l'utilisateur Amazon EC2 Auto Scaling. Pour plus d'informations sur l'utilisation de guillemets avec les paramètres au format JSON, veuillez consulter la rubrique [Quoting Strings](#) dans le Guide de l'utilisateur AWS Command Line Interface.

Exemple 3 : création d'un modèle de lancement qui spécifie le chiffrement des volumes EBS

L'exemple `create-launch-template` suivant crée un modèle de lancement qui inclut des volumes EBS chiffrés créés à partir d'un instantané non chiffré. Il étiquette également les volumes lors de leur création. Si le chiffrement est désactivé par défaut, vous devez spécifier l'option `"Encrypted"` comme indiqué dans l'exemple suivant. Si vous utilisez l'option `"KmsKeyId"` pour spécifier une clé CMK gérée par le client, vous devez également spécifier l'option `"Encrypted"`, même si le chiffrement est activé par défaut.

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

Contenu de `config.json` :

```
{  
  "BlockDeviceMappings":[  
    {  
      "DeviceName":"/dev/sda1",  
      "Ebs":{  
        "VolumeType":"gp2",  
        "DeleteOnTermination":true,  
        "SnapshotId":"snap-066877671789bd71b",  
        "Encrypted":true,  
        "KmsKeyId":"arn:aws:kms:us-east-1:012345678910:key/abcd1234-  
a123-456a-a12b-a123b4cd56ef"  
      }  
    }  
  ],  
  "ImageId":"ami-00068cd7555f543d5",  
  "InstanceType":"c5.large",  
  "TagSpecifications":[  
    {  
      "ResourceType":"volume",  
      "Tags":[  
        {  
          "Key":"encrypted",  
          "Value":"yes"  
        }  
      ]  
    }  
  ]  
}
```

Sortie :

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

Pour de plus amples informations, veuillez consulter la rubrique les rubriques Restauration d'un volume Amazon EBS et Chiffrement par défaut dans le Guide de l'utilisateur Amazon Elastic Compute Cloud.

- Pour plus de détails sur l'API, reportez-vous [CreateLaunchTemplate](#) à la section Référence des AWS CLI commandes.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

- Pour plus de détails sur l'API, voir [CreateLaunchTemplate](#) la section Référence des AWS SDK for JavaScript API.

Python

SDK pour Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Cet exemple crée un modèle de lancement qui inclut un profil d'instance qui accorde des autorisations spécifiques à l'instance, ainsi qu'un script Bash de données utilisateur qui s'exécute sur l'instance après son démarrage.

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

    def __init__(
        self,
        resource_prefix,
        inst_type,
        ami_param,
        autoscaling_client,
        ec2_client,
        ssm_client,
```

```

        iam_client,
    ):
        """
        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is
                created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        self.launch_template_name = f"{resource_prefix}-template"
        self.group_name = f"{resource_prefix}-group"
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"
        self.key_pair_name = f"{resource_prefix}-key-pair"

    def create_template(self, server_startup_script_file, instance_policy_file):
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
        Scaling. The
        launch template specifies a Bash script in its user data field that runs
        after
        the instance is started. This script installs Python packages and starts
        a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
        run
                when an instance starts.

```

```
        :param instance_policy_file: The path to a file that defines a
permissions policy
                                to create and attach to the instance
profile.
:return: Information about the newly created template.
"""
template = {}
try:
    self.create_key_pair(self.key_pair_name)
    self.create_instance_profile(
        instance_policy_file,
        self.instance_policy_name,
        self.instance_role_name,
        self.instance_profile_name,
    )
    with open(server_startup_script_file) as file:
        start_server_script = file.read()
    ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
    ami_id = ami_latest["Parameter"]["Value"]
    lt_response = self.ec2_client.create_launch_template(
        LaunchTemplateName=self.launch_template_name,
        LaunchTemplateData={
            "InstanceType": self.inst_type,
            "ImageId": ami_id,
            "IamInstanceProfile": {"Name": self.instance_profile_name},
            "UserData": base64.b64encode(
                start_server_script.encode(encoding="utf-8")
            ).decode(encoding="utf-8"),
            "KeyName": self.key_pair_name,
        },
    )
    template = lt_response["LaunchTemplate"]
    log.info(
        "Created launch template %s for AMI %s on %s.",
        self.launch_template_name,
        ami_id,
        self.inst_type,
    )
except ClientError as err:
    if (
        err.response["Error"]["Code"]
        == "InvalidLaunchTemplateName.AlreadyExistsException"
    ):
        log.info(
```

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

- Pour plus de détails sur l'API, consultez [CreateLaunchTemplate](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **CreateNetworkAcl** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CreateNetworkAcl`.

CLI

AWS CLI

Pour créer une ACL réseau

Cet exemple crée une ACL réseau pour le VPC spécifié.

Commande :

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

Sortie :

```
{
  "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
  }
}

```

- Pour plus de détails sur l'API, reportez-vous [CreateNetworkAcl](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple crée une ACL réseau pour le VPC spécifié.

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

Sortie :

```

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

```

```
VpcId : vpc-12345678
```

- Pour plus de détails sur l'API, reportez-vous [CreateNetworkAcl](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation `CreateNetworkAclEntry` avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CreateNetworkAclEntry`.

CLI

AWS CLI

Pour créer une entrée ACL réseau

Cet exemple crée une entrée pour l'ACL réseau spécifiée. La règle autorise le trafic entrant depuis n'importe quelle adresse IPv4 (0.0.0.0/0) sur le port UDP 53 (DNS) vers n'importe quel sous-réseau associé. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
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
```

Cet exemple crée une règle pour l'ACL réseau spécifiée qui autorise le trafic entrant depuis n'importe quelle adresse IPv6 (: : /0) sur le port TCP 80 (HTTP).

Commande :

```
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
```

- Pour plus de détails sur l'API, reportez-vous [CreateNetworkAclEntry](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple crée une entrée pour l'ACL réseau spécifiée. La règle autorise le trafic entrant en provenance de n'importe où (0.0.0.0/0) sur le port UDP 53 (DNS) vers n'importe quel sous-réseau associé.

```
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
```

- Pour plus de détails sur l'API, reportez-vous [CreateNetworkAclEntry](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **CreateNetworkInterface** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CreateNetworkInterface`.

CLI

AWS CLI

Exemple 1 : pour spécifier une adresse IPv4 pour une interface réseau

L'`create-network-interface` exemple suivant crée une interface réseau pour le sous-réseau spécifié avec l'adresse IPv4 principale spécifiée.

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

Sortie :

```
{
  "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"
  }
}
```

Exemple 2 : pour créer une interface réseau avec une adresse IPv4 et une adresse IPv6

L'create-network-interfaceexemple suivant crée une interface réseau pour le sous-réseau spécifié avec une adresse IPv4 et une adresse IPv6 sélectionnées par Amazon EC2.

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

```
--groups sg-09dfba7ed20cda78b
```

Sortie :

```
{
  "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"
  }
}
```

Exemple 3 : pour créer une interface réseau avec des options de configuration de suivi des connexions

L'create-network-interfaceexemple suivant crée une interface réseau et configure les délais de suivi des connexions inactives.

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

Sortie :

```
{  
  "NetworkInterface": {  
    "AvailabilityZone": "us-west-2a",  
    "ConnectionTrackingConfiguration": {  
      "TcpEstablishedTimeout": 86400,  
      "UdpTimeout": 60  
    },  
    "Description": "",  
    "Groups": [  
      {  
        "GroupName": "my-security-group",  
        "GroupId": "sg-02e57dbcfe0331c1b"  
      }  
    ],  
    "InterfaceType": "interface",  
    "Ipv6Addresses": [],  
    "MacAddress": "06:4c:53:de:6d:91",  
    "NetworkInterfaceId": "eni-0c133586e08903d0b",  
    "OwnerId": "123456789012",  
    "PrivateDnsName": "ip-10-0-8-94.us-west-2.compute.internal",  
    "PrivateIpAddress": "10.0.8.94",  
    "PrivateIpAddresses": [  
      {  
        "Primary": true,  
        "PrivateDnsName": "ip-10-0-8-94.us-west-2.compute.internal",  
        "PrivateIpAddress": "10.0.8.94"  
      }  
    ],  
    "RequesterId": "AIDA4Z3Y7GSXTMEXAMPLE",  
    "RequesterManaged": false,  
  }  
}
```

```

    "SourceDestCheck": true,
    "Status": "pending",
    "SubnetId": "subnet-00a24d0d67acf6333",
    "TagSet": [],
    "VpcId": "vpc-02723a0feeb9d57b"
  }
}

```

Exemple 4 : pour créer un adaptateur Elastic Fabric

L'create-network-interfaceexemple suivant crée un EFA.

```

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

```

Sortie :

```

{
  "NetworkInterface": {
    "AvailabilityZone": "us-west-2a",
    "Description": "my efa",
    "Groups": [
      {
        "GroupName": "my-efa-sg",
        "GroupId": "sg-02e57dbcfe0331c1b"
      }
    ],
    "InterfaceType": "efa",
    "Ipv6Addresses": [],
    "MacAddress": "06:d7:a4:f7:4d:57",
    "NetworkInterfaceId": "eni-034acc2885e862b65",
    "OwnerId": "123456789012",
    "PrivateDnsName": "ip-10-0-8-180.us-west-2.compute.internal",
    "PrivateIpAddress": "10.0.8.180",
    "PrivateIpAddresses": [
      {
        "Primary": true,
        "PrivateDnsName": "ip-10-0-8-180.us-west-2.compute.internal",
        "PrivateIpAddress": "10.0.8.180"
      }
    ]
  }
}

```

```

    ],
    "RequesterId": "AIDA4Z3Y7GSXTMEXAMPLE",
    "RequesterManaged": false,
    "SourceDestCheck": true,
    "Status": "pending",
    "SubnetId": "subnet-00a24d0d67acf6333",
    "TagSet": [],
    "VpcId": "vpc-02723a0feeeb9d57b"
  }
}

```

Pour plus d'informations, consultez la section [Elastic network interfaces](#) dans le guide de l'utilisateur Amazon EC2.

- Pour plus de détails sur l'API, reportez-vous [CreateNetworkInterface](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple crée l'interface réseau spécifiée.

```

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

```

Sortie :

```

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
PrivateAddresses : {}
RequesterId    :
RequesterManaged : False
SourceDestCheck : True

```

```
Status           : pending
SubnetId         : subnet-1a2b3c4d
TagSet           : {}
VpcId            : vpc-12345678
```

- Pour plus de détails sur l'API, reportez-vous [CreateNetworkInterface](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **CreatePlacementGroup** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CreatePlacementGroup`.

CLI

AWS CLI

Pour créer un groupe de placement

Cet exemple de commande crée un groupe de placement portant le nom spécifié.

Commande :

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

Pour créer un groupe de placement de partitions

Cet exemple de commande crée un groupe de placement de partitions nommé `HDFS-Group-A` avec cinq partitions.

Commande :

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

- Pour plus de détails sur l'API, reportez-vous [CreatePlacementGroup](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple crée un groupe de placement portant le nom spécifié.

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

- Pour plus de détails sur l'API, reportez-vous [CreatePlacementGroup](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **CreateRoute** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CreateRoute`.

CLI

AWS CLI

Pour créer un itinéraire

Cet exemple crée un itinéraire pour la table de routage spécifiée. L'itinéraire correspond à l'ensemble du trafic IPv4 (`0.0.0.0/0`) et l'achemine vers la passerelle Internet spécifiée. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

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

Cet exemple de commande crée une route dans la table de routage `rtb-g8ff4ea2`. L'itinéraire fait correspondre le trafic du bloc CIDR IPv4 `10.0.0.0/16` et l'achemine vers la connexion d'appairage VPC, `pcx-111aaa22`. Cette route permet de diriger le trafic vers le VPC homologue dans la connexion d'appairage VPC. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

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

Cet exemple crée un itinéraire dans la table de routage spécifiée qui correspond à l'ensemble du trafic IPv6 (:::/0) et l'achemine vers la passerelle Internet de sortie uniquement spécifiée.

Commande :

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

- Pour plus de détails sur l'API, reportez-vous [CreateRoute](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple crée l'itinéraire spécifié pour la table de routage spécifiée. L'itinéraire correspond à l'ensemble du trafic et l'envoie à la passerelle Internet spécifiée.

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

Sortie :

```
True
```

- Pour plus de détails sur l'API, reportez-vous [CreateRoute](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **CreateRouteTable** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CreateRouteTable`.

CLI

AWS CLI

Création d'une table de routage

Cet exemple crée une table de routage pour le VPC spécifié.

Commande :

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

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [CreateRouteTable](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple crée une table de routage pour le VPC spécifié.

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

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [CreateRouteTable](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Ruby

Kit SDK pour Ruby

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
require "aws-sdk-ec2"

# Prerequisites:
#
# - A VPC in Amazon VPC.
# - A subnet in that VPC.
# - A gateway attached to that subnet.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param vpc_id [String] The ID of the VPC for the route table.
# @param subnet_id [String] The ID of the subnet for the route table.
# @param gateway_id [String] The ID of the gateway for the route.
# @param destination_cidr_block [String] The destination CIDR block
#   for the route.
# @param tag_key [String] The key portion of the tag for the route table.
# @param tag_value [String] The value portion of the tag for the route table.
# @return [Boolean] true if the route table was created and associated;
#   otherwise, false.
```

```
# @example
#   exit 1 unless route_table_created_and_associated?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     'vpc-0b6f769731EXAMPLE',
#     'subnet-03d9303b57EXAMPLE',
#     'igw-06ca90c011EXAMPLE',
#     '0.0.0.0/0',
#     'my-key',
#     'my-value'
#   )
def route_table_created_and_associated?(
  ec2_resource,
  vpc_id,
  subnet_id,
  gateway_id,
  destination_cidr_block,
  tag_key,
  tag_value
)
  route_table = ec2_resource.create_route_table(vpc_id: vpc_id)
  puts "Created route table with ID '#{route_table.id}'."
  route_table.create_tags(
    tags: [
      {
        key: tag_key,
        value: tag_value
      }
    ]
  )
  puts "Added tags to route table."
  route_table.create_route(
    destination_cidr_block: destination_cidr_block,
    gateway_id: gateway_id
  )
  puts "Created route with destination CIDR block " \
    "'#{destination_cidr_block}' and associated with gateway " \
    "with ID '#{gateway_id}'."
  route_table.associate_with_subnet(subnet_id: subnet_id)
  puts "Associated route table with subnet with ID '#{subnet_id}'."
  return true
rescue StandardError => e
  puts "Error creating or associating route table: #{e.message}"
  puts "If the route table was created but not associated, you should " \
    "clean up by deleting the route table."
```

```
    return false
end

# Example usage:
def run_me
  vpc_id = ""
  subnet_id = ""
  gateway_id = ""
  destination_cidr_block = ""
  tag_key = ""
  tag_value = ""
  region = ""
  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
    puts "Usage: ruby ec2-ruby-example-create-route-table.rb " \
      "VPC_ID SUBNET_ID GATEWAY_ID DESTINATION_CIDR_BLOCK " \
      "TAG_KEY TAG_VALUE REGION"
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  puts "Example: ruby ec2-ruby-example-create-route-table.rb " \
    "vpc-0b6f769731EXAMPLE subnet-03d9303b57EXAMPLE igw-06ca90c011EXAMPLE " \
    "'0.0.0.0/0' my-key my-value us-west-2"
  exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    vpc_id = "vpc-0b6f769731EXAMPLE"
    subnet_id = "subnet-03d9303b57EXAMPLE"
    gateway_id = "igw-06ca90c011EXAMPLE"
    destination_cidr_block = "0.0.0.0/0"
    tag_key = "my-key"
    tag_value = "my-value"
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    region = "us-west-2"
  # Otherwise, use the values as specified at the command prompt.
  else
    vpc_id = ARGV[0]
    subnet_id = ARGV[1]
    gateway_id = ARGV[2]
    destination_cidr_block = ARGV[3]
    tag_key = ARGV[4]
    tag_value = ARGV[5]
    region = ARGV[6]
  end

  ec2_resource = Aws::EC2::Resource.new(region: region)
```

```
if route_table_created_and_associated?(
  ec2_resource,
  vpc_id,
  subnet_id,
  gateway_id,
  destination_cidr_block,
  tag_key,
  tag_value
)
  puts "Route table created and associated."
else
  puts "Route table not created or not associated."
end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Pour plus de détails sur l'API, reportez-vous [CreateRouteTable](#) à la section Référence des AWS SDK for Ruby API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **CreateSecurityGroup** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CreateSecurityGroup`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Démarrer avec les instances](#)

.NET

AWS SDK for .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

    return response.GroupId;
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateSecurityGroup](#) à la section Référence des AWS SDK for .NET API.

Bash

AWS CLI avec le script Bash

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
#####  
# 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
}

```

Fonctions utilitaires utilisées dans cet exemple.

```

#####
# 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
}

```

- Pour plus de détails sur l'API, reportez-vous [CreateSecurityGroup](#) à la section Référence des AWS CLI commandes.

C++

SDK pour C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
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;
```

- Pour plus de détails sur l'API, reportez-vous [CreateSecurityGroup](#) à la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Création d'un groupe de sécurité pour EC2-Classic

Cet exemple crée un groupe de sécurité nommé MySecurityGroup.

Commande :

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

Sortie :

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

Création d'un groupe de sécurité pour EC2-VPC

Cet exemple crée un groupe de sécurité nommé MySecurityGroup pour le VPC spécifié.

Commande :

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

Sortie :

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

Pour plus d'informations, veuillez consulter la rubrique Using Security Groups dans le Guide de l'utilisateur AWS Command Line Interface.

- Pour plus de détails sur l'API, reportez-vous [CreateSecurityGroup](#) à la section Référence des AWS CLI commandes.

Java**SDK pour Java 2.x**** Note**

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

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

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

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

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

    } catch (Ec2Exception e) {
```

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

- Pour plus de détails sur l'API, reportez-vous [CreateSecurityGroup](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

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

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

- Pour plus de détails sur l'API, reportez-vous [CreateSecurityGroup](#) à la section Référence des AWS SDK for JavaScript API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createEC2SecurityGroup(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val resp = ec2.createSecurityGroup(request)
        val ipRange =
            IpRange {
                cidrIp = "0.0.0.0/0"
            }

        val ipPerm =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 80
                fromPort = 80
                ipRanges = listOf(ipRange)
            }

        val ipPerm2 =
```

```
        IpPermission {
            ipProtocol = "tcp"
            toPort = 22
            fromPort = 22
            ipRanges = listOf(ipRange)
        }

        val authRequest =
            AuthorizeSecurityGroupIngressRequest {
                groupName = groupNameVal
                ipPermissions = listOf(ipPerm, ipPerm2)
            }
        ec2.authorizeSecurityGroupIngress(authRequest)
        println("Successfully added ingress policy to Security Group
$groupNameVal")
        return resp.groupId
    }
}
```

- Pour plus de détails sur l'API, consultez [CreateSecurityGroup](#) la section AWS SDK pour la référence de l'API Kotlin.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple crée un groupe de sécurité pour le VPC spécifié.

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

Sortie :

```
sg-12345678
```

Exemple 2 : Cet exemple crée un groupe de sécurité pour EC2-Classique.

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

Sortie :

```
sg-45678901
```

- Pour plus de détails sur l'API, reportez-vous [CreateSecurityGroup](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Python

SDK pour Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

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

    def create(self, group_name, group_description):
```

```
"""
Creates a security group in the default virtual private cloud (VPC) of
the
current account.

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

- Pour plus de détails sur l'API, consultez [CreateSecurityGroup](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
# This code example does the following:
# 1. Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
# 2. Adds inbound rules to the security group.
# 3. Displays information about available security groups.
# 4. Deletes the security group.

require "aws-sdk-ec2"

# Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
#
# Prerequisites:
#
# - A VPC in Amazon Virtual Private Cloud (Amazon VPC).
#
# @param ec2_client [Aws::EC2::Client] An initialized
#   Amazon EC2 client.
# @param group_name [String] A name for the security group.
# @param description [String] A description for the security group.
# @param vpc_id [String] The ID of the VPC for the security group.
# @return [String] The ID of security group that was created.
# @example
#   puts create_security_group(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-security-group',
#     'This is my security group.',
#     'vpc-6713dfEX'
#   )
def create_security_group(
  ec2_client,
  group_name,
  description,
  vpc_id
)
  security_group = ec2_client.create_security_group(
    group_name: group_name,
    description: description,
    vpc_id: vpc_id
  )
  puts "Created security group '#{group_name}' with ID " \
    "'#{security_group.group_id}' in VPC with ID '#{vpc_id}'."
  return security_group.group_id
rescue StandardError => e
```

```
puts "Error creating security group: #{e.message}"
return "Error"
end

# Adds an inbound rule to an Amazon Elastic Compute Cloud (Amazon EC2)
# security group.
#
# Prerequisites:
#
# - The security group.
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @param security_group_id [String] The ID of the security group.
# @param ip_protocol [String] The network protocol for the inbound rule.
# @param from_port [String] The originating port for the inbound rule.
# @param to_port [String] The destination port for the inbound rule.
# @param cidr_ip_range [String] The CIDR IP range for the inbound rule.
# @return
# @example
#   exit 1 unless security_group_ingress_authorized?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'sg-030a858e078f1b9EX',
#     'tcp',
#     '80',
#     '80',
#     '0.0.0.0/0'
#   )
def security_group_ingress_authorized?(
  ec2_client,
  security_group_id,
  ip_protocol,
  from_port,
  to_port,
  cidr_ip_range
)
  ec2_client.authorize_security_group_ingress(
    group_id: security_group_id,
    ip_permissions: [
      {
        ip_protocol: ip_protocol,
        from_port: from_port,
        to_port: to_port,
        ip_ranges: [
          {

```

```

        cidr_ip: cidr_ip_range
      }
    ]
  }
]
)
puts "Added inbound rule to security group '#{security_group_id}' for protocol
" \
  "'#{ip_protocol}' from port '#{from_port}' to port '#{to_port}' " \
  "with CIDR IP range '#{cidr_ip_range}'."
return true
rescue StandardError => e
  puts "Error adding inbound rule to security group: #{e.message}"
  return false
end

# Displays information about a security group's IP permissions set in
# Amazon Elastic Compute Cloud (Amazon EC2).
#
# Prerequisites:
#
# - A security group with inbound rules, outbound rules, or both.
#
# @param p [Aws::EC2::Types::IpPermission] The IP permissions set.
# @example
#   ec2_client = Aws::EC2::Client.new(region: 'us-west-2')
#   response = ec2_client.describe_security_groups
#   unless sg.ip_permissions.empty?
#     describe_security_group_permissions(
#       response.security_groups[0].ip_permissions[0]
#     )
#   end
def describe_security_group_permissions(perm)
  print " Protocol: #{perm.ip_protocol == '-1' ? 'All' : perm.ip_protocol}"

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

unless perm.to_port.nil?

```

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

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

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

print "\n"
end

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

  if response.security_groups.count.positive?
    response.security_groups.each do |sg|
      puts "-" * (sg.group_name.length + 13)
      puts "Name:      #{sg.group_name}"
      puts "Description: #{sg.description}"
      puts "Group ID:    #{sg.group_id}"
      puts "Owner ID:    #{sg.owner_id}"
      puts "VPC ID:      #{sg.vpc_id}"

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

  unless sg.ip_permissions.empty?
```

```

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

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

# Deletes an Amazon Elastic Compute Cloud (Amazon EC2)
# security group.
#
# Prerequisites:
#
# - The security group.
#
# @param ec2_client [Aws::EC2::Client] An initialized
#   Amazon EC2 client.
# @param security_group_id [String] The ID of the security group to delete.
# @return [Boolean] true if the security group was deleted; otherwise, false.
# @example
#   exit 1 unless security_group_deleted?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'sg-030a858e078f1b9EX'
#   )
def security_group_deleted?(ec2_client, security_group_id)
  ec2_client.delete_security_group(group_id: security_group_id)
  puts "Deleted security group '#{security_group_id}'."
  return true
rescue StandardError => e
  puts "Error deleting security group: #{e.message}"
  return false
end
end

```

```
# Example usage:
def run_me
  group_name = ""
  description = ""
  vpc_id = ""
  ip_protocol_http = ""
  from_port_http = ""
  to_port_http = ""
  cidr_ip_range_http = ""
  ip_protocol_ssh = ""
  from_port_ssh = ""
  to_port_ssh = ""
  cidr_ip_range_ssh = ""
  region = ""
  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
    puts "Usage:  ruby ec2-ruby-example-security-group.rb " \
      "GROUP_NAME DESCRIPTION VPC_ID IP_PROTOCOL_1 FROM_PORT_1 TO_PORT_1 " \
      "CIDR_IP_RANGE_1 IP_PROTOCOL_2 FROM_PORT_2 TO_PORT_2 " \
      "CIDR_IP_RANGE_2 REGION"
    puts "Example: ruby ec2-ruby-example-security-group.rb " \
      "my-security-group 'This is my security group.' vpc-6713dfEX " \
      "tcp 80 80 '0.0.0.0/0' tcp 22 22 '0.0.0.0/0' us-west-2"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    group_name = "my-security-group"
    description = "This is my security group."
    vpc_id = "vpc-6713dfEX"
    ip_protocol_http = "tcp"
    from_port_http = "80"
    to_port_http = "80"
    cidr_ip_range_http = "0.0.0.0/0"
    ip_protocol_ssh = "tcp"
    from_port_ssh = "22"
    to_port_ssh = "22"
    cidr_ip_range_ssh = "0.0.0.0/0"
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    region = "us-west-2"
  # Otherwise, use the values as specified at the command prompt.
  else
    group_name = ARGV[0]
    description = ARGV[1]
```

```
vpc_id = ARGV[2]
ip_protocol_http = ARGV[3]
from_port_http = ARGV[4]
to_port_http = ARGV[5]
cidr_ip_range_http = ARGV[6]
ip_protocol_ssh = ARGV[7]
from_port_ssh = ARGV[8]
to_port_ssh = ARGV[9]
cidr_ip_range_ssh = ARGV[10]
region = ARGV[11]
end

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

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

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

  unless security_group_ingress_authorized?(
```

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

- Pour plus de détails sur l'API, reportez-vous [CreateSecurityGroup](#) à la section Référence des AWS SDK for Ruby API.

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

TRY.

```
oo_result = lo_ec2->createsecuritygroup(                                " oo_result is
returned for testing purposes. "
    iv_description = 'Security group example'
    iv_groupname = iv_security_group_name
    iv_vpcid = iv_vpc_id
).
MESSAGE 'Security group created.' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- Pour plus de détails sur l'API, reportez-vous [CreateSecurityGroup](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **CreateSnapshot** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CreateSnapshot`.

CLI

AWS CLI

Pour créer un instantané

Cet exemple de commande crée un instantané du volume avec un ID de volume `vol-1234567890abcdef0` et une brève description pour identifier l'instantané.

Commande :

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

Sortie :

```
{
```

```
"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"
}
```

Pour créer un instantané avec des balises

Cet exemple de commande crée un instantané et applique deux balises : `purpose=prod` et `costcenter=123`.

Commande :

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

Sortie :

```
{
  "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"  
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateSnapshot](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple crée un instantané du volume spécifié.

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

Sortie :

```
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
```

- Pour plus de détails sur l'API, reportez-vous [CreateSnapshot](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation `CreateSpotDatafeedSubscription` avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CreateSpotDatafeedSubscription`.

CLI

AWS CLI

Pour créer un flux de données d'instance Spot

L'exemple suivant crée un flux de données d'instance Spot.

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

Sortie :

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

Le flux de données est stocké dans le compartiment Amazon S3 que vous avez spécifié. Les noms de fichiers de cette source de données sont au format suivant.

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

Pour plus d'informations, consultez le [flux de données des instances Spot](#) dans le guide de l'utilisateur Amazon Elastic Compute Cloud pour les instances Linux.

- Pour plus de détails sur l'API, reportez-vous [CreateSpotDatafeedSubscription](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple crée un flux de données d'instance Spot.

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

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [CreateSpotDatafeedSubscription](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **CreateSubnet** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CreateSubnet`.

CLI

AWS CLI

Exemple 1 : création d'un sous-réseau avec un bloc d'adresses CIDR IPv4 uniquement

L'exemple `create-subnet` suivant crée un sous-réseau dans le VPC spécifié avec le bloc CIDR IPv4 spécifié.

```
aws ec2 create-subnet \  
  --vpc-id vpc-081ec835f3EXAMPLE \  
  --cidr-block 10.0.0.0/24 \  
  --tag-specifications ResourceType=subnet,Tags=[{Key=Name,Value=my-ipv4-only-  
subnet}]
```

Sortie :

```
{
  "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"
  }
}
```

Exemple 2 : création d'un sous-réseau avec des blocs d'adresses CIDR IPv4 et IPv6

L'exemple `create-subnet` suivant crée un sous-réseau dans le VPC spécifié avec les blocs d'adresses CIDR IPv4 et IPv6 spécifiés.

```
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}]
```

Sortie :

```
{
  "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"
  }
}

```

Exemple 3 : création d'un sous-réseau avec un bloc d'adresses CIDR IPv6 uniquement

L'exemple `create-subnet` suivant crée un sous-réseau dans le VPC spécifié avec le bloc CIDR IPv6 spécifié.

```

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}]

```

Sortie :

```
{
  "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"
  }
}
```

Pour plus d'informations, consultez [VPC et sous-réseaux](#) dans le Guide de l'utilisateur Amazon VPC.

- Pour plus de détails sur l'API, reportez-vous [CreateSubnet](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple crée un sous-réseau avec le CIDR spécifié.

```
New-EC2Subnet -VpcId vpc-12345678 -CidrBlock 10.0.0.0/24
```

Sortie :

```
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
```

- Pour plus de détails sur l'API, reportez-vous [CreateSubnet](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Ruby

Kit SDK pour Ruby

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
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}'."
    return true
  rescue StandardError => e
    puts "Error creating or tagging subnet: #{e.message}"
    return false
  end

# Example usage:
def run_me
  vpc_id = ""
  cidr_block = ""
  availability_zone = ""
  tag_key = ""
  tag_value = ""
  region = ""
  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
    puts "Usage:  ruby ec2-ruby-example-create-subnet.rb " \
         "VPC_ID CIDR_BLOCK AVAILABILITY_ZONE TAG_KEY TAG_VALUE REGION"
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts "Example: ruby ec2-ruby-example-create-subnet.rb " \
         "vpc-6713dfEX 10.0.0.0/24 us-west-2a my-key my-value us-west-2"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    vpc_id = "vpc-6713dfEX"
    cidr_block = "10.0.0.0/24"
    availability_zone = "us-west-2a"
    tag_key = "my-key"
    tag_value = "my-value"
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    region = "us-west-2"
  # Otherwise, use the values as specified at the command prompt.
  else
    vpc_id = ARGV[0]
    cidr_block = ARGV[1]
    availability_zone = ARGV[2]
    tag_key = ARGV[3]
    tag_value = ARGV[4]
    region = ARGV[5]
  end
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__
```

- Pour plus de détails sur l'API, reportez-vous [CreateSubnet](#) à la section Référence des AWS SDK for Ruby API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **CreateTags** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CreateTags`.

C++

Kit de développement logiciel (SDK) for C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

```
Aws::EC2::Model::Tag nameTag;
nameTag.SetKey("Name");
nameTag.SetValue(instanceName);

Aws::EC2::Model::CreateTagsRequest createRequest;
createRequest.AddResources(instanceID);
createRequest.AddTags(nameTag);

Aws::EC2::Model::CreateTagsOutcome createOutcome = ec2Client.CreateTags(
    createRequest);
if (!createOutcome.IsSuccess()) {
    std::cerr << "Failed to tag ec2 instance " << instanceID <<
        " with name " << instanceName << ":" <<
        createOutcome.GetError().GetMessage() << std::endl;
    return false;
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateTags](#) à la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Exemple 1 : pour ajouter une balise à une ressource

L'exemple `create-tags` suivant ajoute la balise `Stack=production` à l'image spécifiée ou remplace une balise existante pour l'AMI dont la clé de balise est `Stack`.

```
aws ec2 create-tags \
  --resources ami-1234567890abcdef0 \
  --tags Key=Stack,Value=production
```

Pour plus d'informations, consultez [Ceci est le titre de la rubrique](#) du guide de l'utilisateur Amazon Elastic Compute Cloud pour les instances Linux.

Exemple 2 : pour ajouter des balises à plusieurs ressources

L'exemple `create-tags` suivant ajoute (ou remplace) deux balises pour une AMI et une instance. L'une des balises contient une clé (`webserver`), mais pas de valeur (la valeur

est définie comme une chaîne vide). L'autre balise contient une clé (`stack`) et une valeur (`Production`).

```
aws ec2 create-tags \  
  --resources ami-1a2b3c4d i-1234567890abcdef0 \  
  --tags Key=webserver,Value= Key=stack,Value=Production
```

Pour plus d'informations, consultez [Ceci est le titre de la rubrique](#) du guide de l'utilisateur Amazon Elastic Compute Cloud pour les instances Linux.

Exemple 3 : pour ajouter des balises contenant des caractères spéciaux

L'exemple `create-tags` suivant ajoute la balise `[Group]=test` à une instance. Les crochets (`[` et `]`) sont des caractères spéciaux, qui doivent être échappés. Les exemples suivants utilisent également le caractère de continuation de ligne approprié à chaque environnement.

Si vous utilisez Windows, pour échapper les caractères spéciaux, placez l'élément qui a des caractères spéciaux entre des guillemets doubles (`"`), puis faites précéder chaque guillemet double d'une barre oblique inverse (`\`), comme suit :

```
aws ec2 create-tags ^  
  --resources i-1234567890abcdef0 ^  
  --tags Key=\"[Group]\",Value=test
```

Si vous utilisez Windows PowerShell, entourez l'élément de la valeur comportant des caractères spéciaux de guillemets doubles (`«`), faites précéder chaque guillemet d'une barre oblique inverse (`\`), puis entourez l'ensemble de la structure de clé et de valeur de guillemets simples (`'`) comme suit :

```
aws ec2 create-tags `  
  --resources i-1234567890abcdef0 `  
  --tags 'Key=\"[Group]\",Value=test'
```

Si vous utilisez Linux ou OS X, placez l'élément contenant des caractères spéciaux entre guillemets doubles (`"`), puis placez l'ensemble de la structure de clé et de valeur entre guillemets simples (`'`), comme suit :

```
aws ec2 create-tags \  
  --resources i-1234567890abcdef0 \  
  --tags 'Key="[Group]",Value=test'
```

```
--tags 'Key="[Group]",Value=test'
```

Pour plus d'informations, consultez [Ceci est le titre de la rubrique](#) du guide de l'utilisateur Amazon Elastic Compute Cloud pour les instances Linux.

- Pour plus de détails sur l'API, reportez-vous [CreateTags](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple ajoute une seule balise à la ressource spécifiée. La clé de balise est « MyTag » et la valeur de la balise est « myTagValue ». La syntaxe utilisée dans cet exemple nécessite PowerShell la version 3 ou supérieure.

```
New-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag"; Value="myTagValue" }
```

Exemple 2 : Cet exemple met à jour ou ajoute les balises spécifiées à la ressource spécifiée. La syntaxe utilisée dans cet exemple nécessite PowerShell la version 3 ou supérieure.

```
New-EC2Tag -Resource i-12345678 -Tag @( @{ Key="myTag"; Value="newTagValue" },  
@{ Key="test"; Value="anotherTagValue" } )
```

Exemple 3 : avec PowerShell la version 2, vous devez utiliser New-Object pour créer la balise pour le paramètre Tag.

```
$tag = New-Object Amazon.EC2.Model.Tag  
$tag.Key = "myTag"  
$tag.Value = "myTagValue"  
  
New-EC2Tag -Resource i-12345678 -Tag $tag
```

- Pour plus de détails sur l'API, reportez-vous [CreateTags](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **CreateVolume** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CreateVolume`.

CLI

AWS CLI

Pour créer un volume SSD à usage général (gp2) vide

L'exemple suivant crée un volume SSD à usage général (gp2) de 80 GiB dans la zone de disponibilité spécifiée. Notez que la région actuelle doit être `us-east-1`, ou vous pouvez ajouter le `--region` paramètre pour spécifier la région pour la commande.

```
aws ec2 create-volume \  
  --volume-type gp2 \  
  --size 80 \  
  --availability-zone us-east-1a
```

Sortie :

```
{  
  "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  
}
```

Si vous ne spécifiez aucun type de volume, le type de volume par défaut est `gp2`.

```
aws ec2 create-volume \  
  --size 80 \  
  --availability-zone us-east-1a
```

Exemple 2 : pour créer un volume SSD IOPS provisionné (io1) à partir d'un instantané

L'`create-volume` suivant crée un volume SSD d'IOPS provisionnées (io1) avec 1 000 IOPS provisionnées dans la zone de disponibilité spécifiée à l'aide du snapshot spécifié.

```
aws ec2 create-volume \  
  --volume-type io1 \  
  --iops 1000 \  
  --snapshot-id snap-066877671789bd71b \  
  --availability-zone us-east-1a
```

Sortie :

```
{  
  "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  
}
```

Exemple 3 : pour créer un volume chiffré

L'`create-volume` suivant crée un volume chiffré à l'aide de la clé CMK par défaut pour le chiffrement EBS. Si le chiffrement est désactivé par défaut, vous devez spécifier le `--encrypted` paramètre comme suit.

```
aws ec2 create-volume \  
  --size 80 \  
  --encrypted \  
  --availability-zone us-east-1a
```

Sortie :

```
{  
  "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  
}
```

Si le chiffrement est activé par défaut, l'exemple de commande suivant crée un volume chiffré, même sans le `--encrypted` paramètre.

```
aws ec2 create-volume \  
  --size 80 \  
  --availability-zone us-east-1a
```

Si vous utilisez le `--kms-key-id` paramètre pour spécifier une clé CMK gérée par le client, vous devez le `--encrypted` spécifier même si le chiffrement est activé par défaut.

```
aws ec2 create-volume \  
  --volume-type gp2 \  
  --size 80 \  
  --encrypted \  
  --kms-key-id 0ea3fef3-80a7-4778-9d8c-1c0c6EXAMPLE \  
  --availability-zone us-east-1a
```

Exemple 4 : pour créer un volume avec des balises

L'`create-volume`exemple suivant crée un volume et ajoute deux balises.

```
aws ec2 create-volume \  
  --availability-zone us-east-1a \  
  --volume-type gp2 \  
  --size 80 \  
  --tag-specifications  
  'ResourceType=volume,Tags=[{Key=purpose,Value=production},{Key=cost-  
center,Value=cc123}]'
```

- Pour plus de détails sur l'API, reportez-vous [CreateVolume](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple crée le volume spécifié.

```
New-EC2Volume -Size 50 -AvailabilityZone us-west-2a -VolumeType gp2
```

Sortie :

```
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
```

Exemple 2 : Cet exemple de demande crée un volume et applique une balise avec une clé de pile et une valeur de production.

```
$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
```

- Pour plus de détails sur l'API, reportez-vous [CreateVolume](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation `CreateVpc` avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CreateVpc`.

CLI

AWS CLI

Exemple 1 : création d'un VPC

L'exemple `create-vpc` suivant crée un VPC avec le bloc d'adresses CIDR IPv4 spécifié et une balise `Name`.

```
aws ec2 create-vpc \  
  --cidr-block 10.0.0.0/16 \  
  --tag-specifications ResourceType=vpc,Tags=[{Key=Name,Value=MyVpc}]
```

Sortie :

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

```

    }
  ]
}
}

```

Exemple 2 : création d'un VPC avec une location dédiée

L'exemple `create-vpc` suivant crée un VPC avec le bloc d'adresses CIDR IPv4 spécifié et la location dédiée.

```

aws ec2 create-vpc \
  --cidr-block 10.0.0.0/16 \
  --instance-tenancy dedicated

```

Sortie :

```

{
  "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
  }
}

```

Exemple 3 : création d'un VPC avec un bloc d'adresses CIDR IPv6

L'exemple `create-vpc` suivant crée un VPC avec un bloc d'adresses CIDR IPv6 fourni par Amazon.

```
aws ec2 create-vpc \  
  --cidr-block 10.0.0.0/16 \  
  --amazon-provided-ipv6-cidr-block
```

Sortie :

```
{  
  "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  
  }  
}
```

Exemple 4 : création d'un VPC avec un CIDR depuis un groupe IPAM

L'exemple `create-vpc` suivant crée un VPC avec un CIDR à partir d'un groupe de gestionnaire d'adresses IP (IPAM) Amazon VPC (IPAM).

Linux et 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"}]
```

Sortie :

```
{  
  "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"
}
]
```

Pour plus d'informations, veuillez consulter la rubrique [Création d'un VPC qui utilise un CIDR de groupe IPAM](#) dans le Guide de l'utilisateur Amazon VPC IPAM.

- Pour plus de détails sur l'API, reportez-vous [CreateVpc](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple crée un VPC avec le CIDR spécifié. Amazon VPC crée également les éléments suivants pour le VPC : un ensemble d'options DHCP par défaut, une table de routage principale et une ACL réseau par défaut.

```
New-EC2VPC -CidrBlock 10.0.0.0/16
```

Sortie :

```
CidrBlock      : 10.0.0.0/16
DhcpOptionsId  : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault      : False
State          : pending
Tags           : {}
VpcId          : vpc-12345678
```

- Pour plus de détails sur l'API, reportez-vous [CreateVpc](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Ruby

Kit SDK pour Ruby

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
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}'."
return true
rescue StandardError => e
  puts "#{e.message}"
  return false
end

# Example usage:
def run_me
  cidr_block = ""
  tag_key = ""
  tag_value = ""
  region = ""
  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
    puts "Usage:  ruby ec2-ruby-example-create-vpc.rb " \
        "CIDR_BLOCK TAG_KEY TAG_VALUE REGION"
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts "Example: ruby ec2-ruby-example-create-vpc.rb " \
        "10.0.0.0/24 my-key my-value us-west-2"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    cidr_block = "10.0.0.0/24"
    tag_key = "my-key"
    tag_value = "my-value"
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    region = "us-west-2"
  # Otherwise, use the values as specified at the command prompt.
  else
    cidr_block = ARGV[0]
    tag_key = ARGV[1]
    tag_value = ARGV[2]
    region = ARGV[3]
  end

  ec2_resource = Aws::EC2::Resource.new(region: region)

  if vpc_created_and_tagged?(
    ec2_resource,
```

```
    cidr_block,  
    tag_key,  
    tag_value  
  )  
  puts "VPC created and tagged."  
else  
  puts "VPC not created or not tagged."  
end  
end  
  
run_me if $PROGRAM_NAME == __FILE__
```

- Pour plus de détails sur l'API, reportez-vous [CreateVpc](#) à la section Référence des AWS SDK for Ruby API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **CreateVpcEndpoint** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CreateVpcEndpoint`.

CLI

AWS CLI

Exemple 1 : pour créer un point de terminaison de passerelle

L'`create-vpc-endpoint` exemple suivant crée un point de terminaison VPC de passerelle entre le VPC et Amazon `vpc-1a2b3c4d` S3 dans la `us-east-1` région, et associe une table `rtb-11aa22bb` de routage au point de terminaison.

```
aws ec2 create-vpc-endpoint \  
  --vpc-id vpc-1a2b3c4d \  
  --service-name com.amazonaws.us-east-1.s3 \  
  --route-table-ids rtb-11aa22bb
```

Sortie :

```
{
  "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"
  }
}
```

Pour plus d'informations, consultez la section [Création d'un point de terminaison de passerelle](#) dans le AWS PrivateLink Guide.

Exemple 2 : pour créer un point de terminaison d'interface

L'`create-vpc-endpoint` suivant crée un point de terminaison VPC d'interface entre VPC et Amazon `vpc-1a2b3c4d` S3 dans la région. `us-east-1` La commande crée le point de terminaison dans un sous-réseau `subnet-1a2b3c4d`, l'associe au groupe `sg-1a2b3c4d` de sécurité et ajoute une balise avec la clé « Service » et la valeur « S3 ».

```
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}]
```

Sortie :

```
{
  "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"
}
```

Pour plus d'informations, consultez la section [Création d'un point de terminaison d'interface](#) dans le guide de l'utilisateur pour AWS PrivateLink.

Exemple 3 : pour créer un point de terminaison Gateway Load Balancer

L'commande `aws ec2 create-vpc-endpoint` suivant crée un point de terminaison Gateway Load Balancer entre un VPC `vpc-111122223333aabbcc` et un service configuré à l'aide d'un Gateway Load Balancer.

```
aws ec2 create-vpc-endpoint \  
  --service-name com.amazonaws.vpce.us-east-1.vpce-svc-123123a1c43abc123 \  
  --vpc-endpoint-type GatewayLoadBalancer \  
  --vpc-id vpc-111122223333aabbcc \  
  --subnet-ids subnet-0011aabbcc2233445
```

Sortie :

```
{  
  "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"  
  }  
}
```

Pour plus d'informations, consultez la section [Points de terminaison Gateway Load Balancer](#) dans le guide de l'utilisateur de AWS PrivateLink.

- Pour plus de détails sur l'API, reportez-vous [CreateVpcEndpoint](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple crée un nouveau point de terminaison VPC pour le service `com.amazonaws.eu-west-1.s3` dans le VPC `vpc-0fc1ff23f45b678eb`

```
New-EC2VpcEndpoint -ServiceName com.amazonaws.eu-west-1.s3 -VpcId
vpc-0fc1ff23f45b678eb
```

Sortie :

```
ClientToken VpcEndpoint
-----
Amazon.EC2.Model.VpcEndpoint
```

- Pour plus de détails sur l'API, reportez-vous [CreateVpcEndpoint](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **CreateVpnConnection** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CreateVpnConnection`.

CLI

AWS CLI

Exemple 1 : pour créer une connexion VPN avec un routage dynamique

L'`create-vpn-connection` exemple suivant crée une connexion VPN entre la passerelle privée virtuelle spécifiée et la passerelle client spécifiée, et applique des balises à la connexion VPN. La sortie inclut les informations de configuration de votre dispositif de passerelle client, au format XML.

```
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}]'
```

Sortie :

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

Pour plus d'informations, consultez [Comment fonctionne le VPN de AWS site à site dans le guide de l'utilisateur du VPN](#) de site à site.AWS

Exemple 2 : pour créer une connexion VPN avec un routage statique

L'`create-vpn-connection`exemple suivant crée une connexion VPN entre la passerelle privée virtuelle spécifiée et la passerelle client spécifiée. Les options spécifient le routage

statique. La sortie inclut les informations de configuration de votre dispositif de passerelle client, au format XML.

```
aws ec2 create-vpn-connection \  
  --type ipsec.1 \  
  --customer-gateway-id cgw-001122334455aabbcc \  
  --vpn-gateway-id vgw-1a1a1a1a1a1a2b2b2 \  
  --options "{\"StaticRoutesOnly\":true}"
```

Sortie :

```
{  
  "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": []  
  }  
}
```

Pour plus d'informations, consultez [Comment fonctionne le VPN de AWS site à site dans le guide de l'utilisateur du VPN](#) de site à site.AWS

Exemple 3 : pour créer une connexion VPN et spécifier votre propre CIDR interne et votre clé pré-partagée

L'create-vpn-connectionexemple suivant crée une connexion VPN et spécifie le bloc CIDR de l'adresse IP interne et une clé pré-partagée personnalisée pour chaque tunnel. Les valeurs spécifiées sont renvoyées dans les CustomerGatewayConfiguration informations.

```
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}]'
```

Sortie :

```
{  
  "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": []
  }
}

```

Pour plus d'informations, consultez [Comment fonctionne le VPN de AWS site à site dans le guide de l'utilisateur du VPN](#) de site à site.AWS

Exemple 4 : pour créer une connexion VPN prenant en charge le trafic IPv6

L'create-vpn-connectionexemple suivant crée une connexion VPN qui prend en charge le trafic IPv6 entre la passerelle de transit spécifiée et la passerelle client spécifiée. Les options de tunnel pour les deux tunnels spécifient que AWS doit lancer la négociation IKE.

```

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}]

```

Sortie :

```

{
  "VpnConnection": {
    "CustomerGatewayConfiguration": "..configuration information...",
    "CustomerGatewayId": "cgw-001122334455aabbcc",
    "Category": "VPN",
    "State": "pending",
    "VpnConnectionId": "vpn-11111111122222222",
    "TransitGatewayId": "tgw-12312312312312312",
    "Options": {
      "EnableAcceleration": false,
      "StaticRoutesOnly": false,
      "LocalIpv6NetworkCidr": "::/0",
      "RemoteIpv6NetworkCidr": "::/0",
      "TunnelInsideIpVersion": "ipv6",
      "TunnelOptions": [
        {
          "OutsideIpAddress": "203.0.113.3",
          "StartupAction": "start"
        }
      ],
    }
  }
}

```

```

        {
            "OutsideIpAddress": "203.0.113.5",
            "StartupAction": "start"
        }
    ],
    "Routes": [],
    "Tags": []
}
}

```

Pour plus d'informations, consultez [Comment fonctionne le VPN de AWS site à site dans le guide de l'utilisateur du VPN](#) de site à site.AWS

- Pour plus de détails sur l'API, reportez-vous [CreateVpnConnection](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple crée une connexion VPN entre la passerelle privée virtuelle spécifiée et la passerelle client spécifiée. La sortie inclut les informations de configuration dont votre administrateur réseau a besoin, au format XML.

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId vgw-1a2b3c4d
```

Sortie :

```

CustomerGatewayConfiguration : [XML document]
CustomerGatewayId           : cgw-1a2b3c4d
Options                     :
Routes                     : {}
State                       : pending
Tags                       : {}
Type                       :
VgwTelemetry               : {}
VpnConnectionId           : vpn-12345678
VpnGatewayId              : vgw-1a2b3c4d

```

Exemple 2 : Cet exemple crée la connexion VPN et capture la configuration dans un fichier portant le nom spécifié.

```
(New-EC2VpnConnection -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId  
vgw-1a2b3c4d).CustomerGatewayConfiguration | Out-File C:\path\vpn-  
configuration.xml
```

Exemple 3 : Cet exemple crée une connexion VPN, avec un routage statique, entre la passerelle privée virtuelle spécifiée et la passerelle client spécifiée.

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId  
vgw-1a2b3c4d -Options_StaticRoutesOnly $true
```

- Pour plus de détails sur l'API, reportez-vous [CreateVpnConnection](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **CreateVpnConnectionRoute** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CreateVpnConnectionRoute`.

CLI

AWS CLI

Pour créer un itinéraire statique pour une connexion VPN

Cet exemple crée un itinéraire statique pour la connexion VPN spécifiée. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 create-vpn-connection-route --vpn-connection-id vpn-40f41529 --  
destination-cidr-block 11.12.0.0/16
```

- Pour plus de détails sur l'API, reportez-vous [CreateVpnConnectionRoute](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple crée la route statique spécifiée pour la connexion VPN spécifiée.

```
New-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock 11.12.0.0/16
```

- Pour plus de détails sur l'API, reportez-vous [CreateVpnConnectionRoute](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation `CreateVpnGateway` avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `CreateVpnGateway`.

CLI

AWS CLI

Pour créer une passerelle privée virtuelle

Cet exemple crée une passerelle privée virtuelle.

Commande :

```
aws ec2 create-vpn-gateway --type ipsec.1
```

Sortie :

```
{
  "VpnGateway": {
    "AmazonSideAsn": 64512,
    "State": "available",
    "Type": "ipsec.1",
    "VpnGatewayId": "vgw-9a4cacf3",
    "VpcAttachments": []
  }
}
```

```
}  
}
```

Pour créer une passerelle privée virtuelle avec un ASN spécifique côté Amazon

Cet exemple crée une passerelle privée virtuelle et spécifie le numéro de système autonome (ASN) pour le côté Amazon de la session BGP.

Commande :

```
aws ec2 create-vpn-gateway --type ipsec.1 --amazon-side-asn 65001
```

Sortie :

```
{  
  "VpnGateway": {  
    "AmazonSideAsn": 65001,  
    "State": "available",  
    "Type": "ipsec.1",  
    "VpnGatewayId": "vgw-9a4cacf3",  
    "VpcAttachments": []  
  }  
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateVpnGateway](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple crée la passerelle privée virtuelle spécifiée.

```
New-EC2VpnGateway -Type ipsec.1
```

Sortie :

```
AvailabilityZone :  
State           : available  
Tags            : {}  
Type            : ipsec.1
```

```
VpcAttachments    : {}  
VpnGatewayId     : vgw-1a2b3c4d
```

- Pour plus de détails sur l'API, reportez-vous [CreateVpnGateway](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation `DeleteCustomerGateway` avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DeleteCustomerGateway`.

CLI

AWS CLI

Pour supprimer une passerelle client

Cet exemple supprime la passerelle client spécifiée. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 delete-customer-gateway --customer-gateway-id cgw-0e11f167
```

- Pour plus de détails sur l'API, reportez-vous [DeleteCustomerGateway](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la passerelle client spécifiée. Vous êtes invité à confirmer avant de poursuivre l'opération, sauf si vous spécifiez également le paramètre `Force`.

```
Remove-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

Sortie :

```
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"):
```

- Pour plus de détails sur l'API, reportez-vous [DeleteCustomerGateway](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DeleteDhcpOptions** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DeleteDhcpOptions`.

CLI

AWS CLI

Pour supprimer un ensemble d'options DHCP

Cet exemple supprime le jeu d'options DHCP spécifié. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 delete-dhcp-options --dhcp-options-id dopt-d9070ebb
```

- Pour plus de détails sur l'API, reportez-vous [DeleteDhcpOptions](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : cet exemple supprime le jeu d'options DHCP spécifié. Vous êtes invité à confirmer avant de poursuivre l'opération, sauf si vous spécifiez également le paramètre `Force`.

```
Remove-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d
```

Sortie :

```
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"):
```

- Pour plus de détails sur l'API, reportez-vous [DeleteDhcpOptions](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DeleteFlowLogs** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DeleteFlowLogs`.

CLI

AWS CLI

Pour supprimer un journal de flux

L'`delete-flow-log` exemple suivant supprime le journal de flux spécifié.

```
aws ec2 delete-flow-logs --flow-log-id fl-11223344556677889
```

Sortie :

```
{
  "Unsuccessful": []
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteFlowLogs](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le fichier FlowLogId fl-01a2b3456a789c01 donné

```
Remove-EC2FlowLog -FlowLogId fl-01a2b3456a789c01
```

Sortie :

```
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
```

- Pour plus de détails sur l'API, reportez-vous [DeleteFlowLogs](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DeleteInternetGateway** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DeleteInternetGateway`.

CLI

AWS CLI

Pour supprimer une passerelle Internet

L'`delete-internet-gateway` exemple suivant supprime la passerelle Internet spécifiée.

```
aws ec2 delete-internet-gateway \
  --internet-gateway-id igw-0d0fb496b3EXAMPLE
```

Cette commande ne produit aucun résultat.

Pour plus d'informations, consultez [Passerelles Internet](#) dans le Guide de l'utilisateur Amazon VPC.

- Pour plus de détails sur l'API, reportez-vous [DeleteInternetGateway](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : cet exemple supprime la passerelle Internet spécifiée. Vous êtes invité à confirmer avant de poursuivre l'opération, sauf si vous spécifiez également le paramètre Force.

```
Remove-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

Sortie :

```
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"):
```

- Pour plus de détails sur l'API, reportez-vous [DeleteInternetGateway](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DeleteKeyPair** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DeleteKeyPair`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Démarrer avec les instances](#)

.NET

AWS SDK for .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Delete an Amazon EC2 key pair.
/// </summary>
/// <param name="keyPairName">The name of the key pair to delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteKeyPair(string keyPairName)
{
    try
    {
        await _amazonEC2.DeleteKeyPairAsync(new
DeleteKeyPairRequest(keyPairName)).ConfigureAwait(false);
        return true;
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Couldn't delete the key pair because:
{ex.Message}");
        return false;
    }
}

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

- Pour plus de détails sur l'API, reportez-vous [DeleteKeyPair](#) à la section Référence des AWS SDK for .NET API.

Bash

AWS CLI avec le script Bash

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

    # Retrieve the calling parameters.
    while getopt "n:h" option; do
```

```

    case "${option}" in
        n) key_pair_name="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

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

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

return 0
}

```

Fonctions utilitaires utilisées dans cet exemple.

```

#####
# 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
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteKeyPair](#) à la section Référence des AWS CLI commandes.

C++

SDK pour C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
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;
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteKeyPair](#) à la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Suppression d'une paire de clés

L'`delete-key-pair`exemple suivant supprime la paire de clés spécifiée.

```
aws ec2 delete-key-pair \
    --key-name my-key-pair
```

Sortie :

```
{
  "Return": true,
  "KeyPairId": "key-03c8d3aceb53b507"
}
```

Pour plus d'informations, voir [Création et suppression de paires de clés](#) dans le Guide de l'utilisateur de l'interface de ligne de commande AWS.

- Pour plus de détails sur l'API, reportez-vous [DeleteKeyPair](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void deleteKeys(Ec2Client ec2, String keyPair) {
    try {
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
            .keyName(keyPair)
            .build();

        ec2.deleteKeyPair(request);
        System.out.println("Successfully deleted key pair named " + keyPair);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteKeyPair](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

export const main = async () => {
  const command = new DeleteKeyPairCommand({
    KeyName: "KEY_PAIR_NAME",
  });

  try {
    await client.send(command);
    console.log("Successfully deleted key pair.");
  } catch (err) {
    console.error(err);
  }
};
```

- Pour plus de détails sur l'API, reportez-vous [DeleteKeyPair](#) à la section Référence des AWS SDK for JavaScript API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteKeys(keyPair: String?) {
    val request =
        DeleteKeyPairRequest {
            keyName = keyPair
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.deleteKeyPair(request)
        println("Successfully deleted key pair named $keyPair")
    }
}
```

- Pour plus de détails sur l'API, consultez [DeleteKeyPair](#) la section AWS SDK pour la référence de l'API Kotlin.

PowerShell

Outils pour PowerShell

Exemple 1 : cet exemple supprime la paire de clés spécifiée. Vous êtes invité à confirmer avant de poursuivre l'opération, sauf si vous spécifiez également le paramètre Force.

```
Remove-EC2KeyPair -KeyName my-key-pair
```

Sortie :

```
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"):
```

- Pour plus de détails sur l'API, reportez-vous [DeleteKeyPair](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Python

SDK pour Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

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

    def delete(self):
        """
        Deletes a key pair.
        """
        if self.key_pair is None:
```

```
        logger.info("No key pair to delete.")
        return

    key_name = self.key_pair.name
    try:
        self.key_pair.delete()
        self.key_pair = None
    except ClientError as err:
        logger.error(
            "Couldn't delete key %s. Here's why: %s : %s",
            key_name,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

- Pour plus de détails sur l'API, consultez [DeleteKeyPair](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
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.
```

- Pour plus de détails sur l'API, reportez-vous [DeleteKeyPair](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DeleteLaunchTemplate** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DeleteLaunchTemplate`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Créer et gérer un service résilient](#)

.NET

AWS SDK for .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Delete a launch template by name.
/// </summary>
/// <param name="templateName">The name of the template to delete.</param>
/// <returns>Async task.</returns>
public async Task DeleteTemplateByName(string templateName)
{
    try
    {
        await _amazonEc2.DeleteLaunchTemplateAsync(
            new DeleteLaunchTemplateRequest()
            {
```

```
        LaunchTemplateName = templateName
    });
}
catch (AmazonClientException)
{
    Console.WriteLine($"Unable to delete template {templateName}.");
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteLaunchTemplate](#) à la section Référence des AWS SDK for .NET API.

CLI

AWS CLI

Suppression d'un modèle de lancement

Cet exemple supprime le modèle de lancement spécifié.

Commande :

```
aws ec2 delete-launch-template --launch-template-id lt-0abcd290751193123
```

Sortie :

```
{
  "LaunchTemplate": {
    "LatestVersionNumber": 2,
    "LaunchTemplateId": "lt-0abcd290751193123",
    "LaunchTemplateName": "TestTemplate",
    "DefaultVersionNumber": 2,
    "CreatedBy": "arn:aws:iam::123456789012:root",
    "CreateTime": "2017-11-23T16:46:25.000Z"
  }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteLaunchTemplate](#) à la section Référence des AWS CLI commandes.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
await client.send(  
  new DeleteLaunchTemplateCommand({  
    LaunchTemplateName: NAMES.launchTemplateName,  
  }),  
);
```

- Pour plus de détails sur l'API, reportez-vous [DeleteLaunchTemplate](#) à la section Référence des AWS SDK for JavaScript API.

Python

SDK pour Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class AutoScaler:  
    """  
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.  
    """  
  
    def __init__(  
        self,  
        resource_prefix,  
        inst_type,  
        ami_param,
```

```
        autoscaling_client,  
        ec2_client,  
        ssm_client,  
        iam_client,  
    ):  
        """  
        :param resource_prefix: The prefix for naming AWS resources that are  
created by this class.  
        :param inst_type: The type of EC2 instance to create, such as t3.micro.  
        :param ami_param: The Systems Manager parameter used to look up the AMI  
that is  
            created.  
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.  
        :param ec2_client: A Boto3 EC2 client.  
        :param ssm_client: A Boto3 Systems Manager client.  
        :param iam_client: A Boto3 IAM client.  
        """  
        self.inst_type = inst_type  
        self.ami_param = ami_param  
        self.autoscaling_client = autoscaling_client  
        self.ec2_client = ec2_client  
        self.ssm_client = ssm_client  
        self.iam_client = iam_client  
        self.launch_template_name = f"{resource_prefix}-template"  
        self.group_name = f"{resource_prefix}-group"  
        self.instance_policy_name = f"{resource_prefix}-pol"  
        self.instance_role_name = f"{resource_prefix}-role"  
        self.instance_profile_name = f"{resource_prefix}-prof"  
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"  
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"  
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"  
        self.key_pair_name = f"{resource_prefix}-key-pair"  
  
    def delete_template(self):  
        """  
        Deletes a launch template.  
        """  
        try:  
            self.ec2_client.delete_launch_template(  
                LaunchTemplateName=self.launch_template_name  
            )  
            self.delete_instance_profile(  
                self.instance_profile_name, self.instance_role_name
```

```
    )
    log.info("Launch template %s deleted.", self.launch_template_name)
except ClientError as err:
    if (
        err.response["Error"]["Code"]
        == "InvalidLaunchTemplateName.NotFoundException"
    ):
        log.info(
            "Launch template %s does not exist, nothing to do.",
            self.launch_template_name,
        )
    else:
        raise AutoScalerError(
            f"Couldn't delete launch template
{self.launch_template_name}: {err}."
        )
```

- Pour plus de détails sur l'API, consultez [DeleteLaunchTemplate](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DeleteNetworkAcl** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DeleteNetworkAcl`.

CLI

AWS CLI

Pour supprimer une ACL réseau

Cet exemple supprime l'ACL réseau spécifiée. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 delete-network-acl --network-acl-id acl-5fb85d36
```

- Pour plus de détails sur l'API, reportez-vous [DeleteNetworkAcl](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple supprime l'ACL réseau spécifiée. Vous êtes invité à confirmer avant de poursuivre l'opération, sauf si vous spécifiez également le paramètre Force.

```
Remove-EC2NetworkAcl -NetworkAclId acl-12345678
```

Sortie :

```
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"):
```

- Pour plus de détails sur l'API, reportez-vous [DeleteNetworkAcl](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DeleteNetworkAclEntry** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DeleteNetworkAclEntry`.

CLI

AWS CLI

Pour supprimer une entrée ACL réseau

Cet exemple supprime la règle d'entrée numéro 100 de l'ACL réseau spécifiée. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 delete-network-acl-entry --network-acl-id acl-5fb85d36 --ingress --rule-number 100
```

- Pour plus de détails sur l'API, reportez-vous [DeleteNetworkAclEntry](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la règle spécifiée de l'ACL réseau spécifiée. Vous êtes invité à confirmer avant de poursuivre l'opération, sauf si vous spécifiez également le paramètre Force.

```
Remove-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100
```

Sortie :

```
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"):
```

- Pour plus de détails sur l'API, reportez-vous [DeleteNetworkAclEntry](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DeleteNetworkInterface** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DeleteNetworkInterface`.

CLI

AWS CLI

Pour supprimer une interface réseau

Cet exemple supprime l'interface réseau spécifiée. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 delete-network-interface --network-interface-id eni-e5aa89a3
```

- Pour plus de détails sur l'API, reportez-vous [DeleteNetworkInterface](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : cet exemple supprime l'interface réseau spécifiée. Vous êtes invité à confirmer avant de poursuivre l'opération, sauf si vous spécifiez également le paramètre Force.

```
Remove-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

Sortie :

```
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"):
```

- Pour plus de détails sur l'API, reportez-vous [DeleteNetworkInterface](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation `DeletePlacementGroup` avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DeletePlacementGroup`.

CLI

AWS CLI

Pour supprimer un groupe de placement

Cet exemple de commande supprime le groupe de placement spécifié.

Commande :

```
aws ec2 delete-placement-group --group-name my-cluster
```

- Pour plus de détails sur l'API, reportez-vous [DeletePlacementGroup](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : cet exemple supprime le groupe de placement spécifié. Vous êtes invité à confirmer avant de poursuivre l'opération, sauf si vous spécifiez également le paramètre `Force`.

```
Remove-EC2PlacementGroup -GroupName my-placement-group
```

Sortie :

```
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"):
```

- Pour plus de détails sur l'API, reportez-vous [DeletePlacementGroup](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DeleteRoute** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DeleteRoute`.

CLI

AWS CLI

Pour supprimer un itinéraire

Cet exemple supprime l'itinéraire spécifié de la table de routage spécifiée. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 delete-route --route-table-id rtb-22574640 --destination-cidr-block
0.0.0.0/0
```

- Pour plus de détails sur l'API, reportez-vous [DeleteRoute](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple supprime l'itinéraire spécifié de la table de routage spécifiée. Vous êtes invité à confirmer avant de poursuivre l'opération, sauf si vous spécifiez également le paramètre `Force`.

```
Remove-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0
```

Sortie :

```
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"):
```

- Pour plus de détails sur l'API, reportez-vous [DeleteRoute](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DeleteRouteTable** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DeleteRouteTable`.

CLI

AWS CLI

Pour supprimer une table de routage

Cet exemple supprime la table de routage spécifiée. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 delete-route-table --route-table-id rtb-22574640
```

- Pour plus de détails sur l'API, reportez-vous [DeleteRouteTable](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la table de routage spécifiée. Vous êtes invité à confirmer avant de poursuivre l'opération, sauf si vous spécifiez également le paramètre `Force`.

```
Remove-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

Sortie :

```
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"):
```

- Pour plus de détails sur l'API, reportez-vous [DeleteRouteTable](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DeleteSecurityGroup** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DeleteSecurityGroup`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Démarrer avec les instances](#)

.NET

AWS SDK for .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

```
public async Task<bool> DeleteSecurityGroup(string groupId)
{
    var response = await _amazonEC2.DeleteSecurityGroupAsync(new
DeleteSecurityGroupRequest { GroupId = groupId });
    return response.HttpStatusCode == HttpStatusCode.OK;
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteSecurityGroup](#) à la section Référence des AWS SDK for .NET API.

Bash

AWS CLI avec le script Bash

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
#####
# 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
}

```

Fonctions utilitaires utilisées dans cet exemple.

```

#####
# 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
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteSecurityGroup](#) à la section Référence des AWS CLI commandes.

C++

SDK pour C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::DeleteSecurityGroupRequest request;

request.SetGroupId(securityGroupID);
auto outcome = ec2Client.DeleteSecurityGroup(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Failed to delete security group " << securityGroupID <<
        ":" << outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Successfully deleted security group " << securityGroupID <<
        std::endl;
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteSecurityGroup](#) à la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

[EC2-Classique] Suppression d'un groupe de sécurité

Cet exemple supprime le groupe de sécurité nommé MySecurityGroup. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 delete-security-group --group-name MySecurityGroup
```

[EC2-VPC] Suppression d'un groupe de sécurité

Cet exemple supprime le groupe de sécurité dont l'ID est `sg-903004f8`. Notez que vous ne pouvez pas référencer un groupe de sécurité pour EC2-VPC par nom. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 delete-security-group --group-id sg-903004f8
```

Pour plus d'informations, veuillez consulter la rubrique [Using Security Groups](#) dans le Guide de l'utilisateur AWS Command Line Interface.

- Pour plus de détails sur l'API, reportez-vous [DeleteSecurityGroup](#) à la section Référence des AWS CLI commandes.

Java**SDK pour Java 2.x**** Note**

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void deleteEC2SecGroup(Ec2Client ec2, String groupId) {
    try {
        DeleteSecurityGroupRequest request =
DeleteSecurityGroupRequest.builder()
        .groupId(groupId)
        .build();

        ec2.deleteSecurityGroup(request);
        System.out.println("Successfully deleted security group with Id " +
groupId);
    }
}
```

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

- Pour plus de détails sur l'API, reportez-vous [DeleteSecurityGroup](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import { DeleteSecurityGroupCommand } from "@aws-sdk/client-ec2";  
  
import { client } from "../libs/client.js";  
  
export const main = async () => {  
    const command = new DeleteSecurityGroupCommand({  
        GroupId: "GROUP_ID",  
    });  
  
    try {  
        await client.send(command);  
        console.log("Security group deleted successfully.");  
    } catch (err) {  
        console.error(err);  
    }  
};
```

- Pour plus de détails sur l'API, reportez-vous [DeleteSecurityGroup](#) à la section Référence des AWS SDK for JavaScript API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteEC2SecGroup(groupIdVal: String) {
    val request =
        DeleteSecurityGroupRequest {
            groupId = groupIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.deleteSecurityGroup(request)
        println("Successfully deleted Security Group with id $groupIdVal")
    }
}
```

- Pour plus de détails sur l'API, consultez [DeleteSecurityGroup](#) la section AWS SDK pour la référence de l'API Kotlin.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le groupe de sécurité spécifié pour EC2-VPC. Vous êtes invité à confirmer avant de poursuivre l'opération, sauf si vous spécifiez également le paramètre Force.

```
Remove-EC2SecurityGroup -GroupId sg-12345678
```

Sortie :

```
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"):
```

Exemple 2 : Cet exemple supprime le groupe de sécurité spécifié pour EC2-Classique.

```
Remove-EC2SecurityGroup -GroupName my-security-group -Force
```

- Pour plus de détails sur l'API, reportez-vous [DeleteSecurityGroup](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Python

SDK pour Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

    @classmethod
    def from_resource(cls):
```

```
ec2_resource = boto3.resource("ec2")
return cls(ec2_resource)

def delete(self):
    """
    Deletes the security group.
    """
    if self.security_group is None:
        logger.info("No security group to delete.")
        return

    group_id = self.security_group.id
    try:
        self.security_group.delete()
    except ClientError as err:
        logger.error(
            "Couldn't delete security group %s. Here's why: %s: %s",
            group_id,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

- Pour plus de détails sur l'API, consultez [DeleteSecurityGroup](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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.
```

- Pour plus de détails sur l'API, reportez-vous [DeleteSecurityGroup](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DeleteSnapshot** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DeleteSnapshot`.

CLI

AWS CLI

Suppression d'un instantané

Cet exemple de commande supprime un instantané dont l'ID de capture est `snap-1234567890abcdef0`. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 delete-snapshot --snapshot-id snap-1234567890abcdef0
```

- Pour plus de détails sur l'API, reportez-vous [DeleteSnapshot](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le cliché spécifié. Vous êtes invité à confirmer avant de poursuivre l'opération, sauf si vous spécifiez également le paramètre `Force`.

```
Remove-EC2Snapshot -SnapshotId snap-12345678
```

Sortie :

```
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"):
```

- Pour plus de détails sur l'API, reportez-vous [DeleteSnapshot](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Rust

SDK pour Rust

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
async fn delete_snapshot(client: &Client, id: &str) -> Result<(), Error> {
    client.delete_snapshot().snapshot_id(id).send().await?;

    println!("Deleted");

    Ok(())
}
```

- Pour plus de détails sur l'API, voir [DeleteSnapshot](#) la section de référence de l'API AWS SDK for Rust.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation `DeleteSpotDatafeedSubscription` avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DeleteSpotDatafeedSubscription`.

CLI

AWS CLI

Pour annuler un abonnement au flux de données d'une instance Spot

Cet exemple de commande supprime un abonnement à un flux de données Spot pour le compte. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 delete-spot-datafeed-subscription
```

- Pour plus de détails sur l'API, reportez-vous [DeleteSpotDatafeedSubscription](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le flux de données de votre instance Spot. Vous êtes invité à confirmer avant de poursuivre l'opération, sauf si vous spécifiez également le paramètre `Force`.

```
Remove-EC2SpotDatafeedSubscription
```

Sortie :

```
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"):
```

- Pour plus de détails sur l'API, reportez-vous [DeleteSpotDatafeedSubscription](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DeleteSubnet** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DeleteSubnet`.

CLI

AWS CLI

Pour supprimer un sous-réseau

Cet exemple supprime le sous-réseau spécifié. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 delete-subnet --subnet-id subnet-9d4a7b6c
```

- Pour plus de détails sur l'API, reportez-vous [DeleteSubnet](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le sous-réseau spécifié. Vous êtes invité à confirmer avant de poursuivre l'opération, sauf si vous spécifiez également le paramètre `Force`.

```
Remove-EC2Subnet -SubnetId subnet-1a2b3c4d
```

Sortie :

```
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"):
```

- Pour plus de détails sur l'API, reportez-vous [DeleteSubnet](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DeleteTags** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DeleteTags`.

CLI

AWS CLI

Exemple 1 : pour supprimer un tag d'une ressource

L'`delete-tag` exemple suivant supprime le tag `Stack=Test` de l'image spécifiée. Lorsque vous spécifiez à la fois une valeur et un nom de clé, la balise est supprimée uniquement si la valeur de la balise correspond à la valeur spécifiée.

```
aws ec2 delete-tags \  
  --resources ami-1234567890abcdef0 \  
  --tags Key=Stack,Value=Test
```

Il est facultatif de spécifier la valeur d'une balise. L'`delete-tag` exemple suivant supprime la balise portant le nom `purpose` de la clé de l'instance spécifiée, quelle que soit la valeur de la balise.

```
aws ec2 delete-tags \  
  --resources i-1234567890abcdef0 \  
  --tags Key=purpose
```

Si vous spécifiez la chaîne vide comme valeur de balise, la balise est supprimée uniquement si la valeur de la balise est la chaîne vide. L'exemple suivant indique la chaîne vide comme valeur de balise pour la balise à supprimer.

```
aws ec2 delete-tags \  
  --resources i-1234567890abcdef0 \  
  --tags Key=Name,Value=
```

Exemple 2 : pour supprimer un tag de plusieurs ressources

L'exemple suivant supprime le tag ``purpose=test`` à la fois d'une instance et d'une AMI. Comme indiqué dans l'exemple précédent, vous pouvez omettre la valeur du tag dans la commande.

```
aws ec2 delete-tags \  
  --resources i-1234567890abcdef0 ami-1234567890abcdef0 \  
  --tags Key=Purpose
```

- Pour plus de détails sur l'API, reportez-vous [DeleteTags](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la balise spécifiée de la ressource spécifiée, quelle que soit la valeur de la balise. La syntaxe utilisée dans cet exemple nécessite PowerShell la version 3 ou ultérieure.

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag" } -Force
```

Exemple 2 : Cet exemple supprime la balise spécifiée de la ressource spécifiée, mais uniquement si la valeur de la balise correspond. La syntaxe utilisée dans cet exemple nécessite PowerShell la version 3 ou ultérieure.

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag";Value="myTagValue" } -Force
```

Exemple 3 : Cet exemple supprime la balise spécifiée de la ressource spécifiée, quelle que soit la valeur de la balise.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

Exemple 4 : Cet exemple supprime la balise spécifiée de la ressource spécifiée, mais uniquement si la valeur de la balise correspond.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

- Pour plus de détails sur l'API, reportez-vous [DeleteTags](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DeleteVolume** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DeleteVolume`.

CLI

AWS CLI

Pour supprimer un volume

Cet exemple de commande supprime un volume disponible dont l'ID de volume est `vol-049df61146c4d7901`. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 delete-volume --volume-id vol-049df61146c4d7901
```

- Pour plus de détails sur l'API, reportez-vous [DeleteVolume](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : cet exemple détache le volume spécifié. Vous êtes invité à confirmer avant de poursuivre l'opération, sauf si vous spécifiez également le paramètre Force.

```
Remove-EC2Volume -VolumeId vol-12345678
```

Sortie :

```
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"):
```

- Pour plus de détails sur l'API, reportez-vous [DeleteVolume](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DeleteVpc** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DeleteVpc`.

CLI

AWS CLI

Pour supprimer un VPC

Cet exemple supprime le VPC spécifié. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 delete-vpc --vpc-id vpc-a01106c2
```

- Pour plus de détails sur l'API, reportez-vous [DeleteVpc](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le VPC spécifié. Vous êtes invité à confirmer avant de poursuivre l'opération, sauf si vous spécifiez également le paramètre Force.

```
Remove-EC2Vpc -VpcId vpc-12345678
```

Sortie :

```
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"):
```

- Pour plus de détails sur l'API, reportez-vous [DeleteVpc](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DeleteVpnConnection** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DeleteVpnConnection`.

CLI

AWS CLI

Pour supprimer une connexion VPN

Cet exemple supprime la connexion VPN spécifiée. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 delete-vpn-connection --vpn-connection-id vpn-40f41529
```

- Pour plus de détails sur l'API, reportez-vous [DeleteVpnConnection](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la connexion VPN spécifiée. Vous êtes invité à confirmer avant de poursuivre l'opération, sauf si vous spécifiez également le paramètre Force.

```
Remove-EC2VpnConnection -VpnConnectionId vpn-12345678
```

Sortie :

```
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"):
```

- Pour plus de détails sur l'API, reportez-vous [DeleteVpnConnection](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation `DeleteVpnConnectionRoute` avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DeleteVpnConnectionRoute`.

CLI

AWS CLI

Pour supprimer un itinéraire statique d'une connexion VPN

Cet exemple supprime la route statique spécifiée de la connexion VPN spécifiée. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 delete-vpn-connection-route --vpn-connection-id vpn-40f41529 --
destination-cidr-block 11.12.0.0/16
```

- Pour plus de détails sur l'API, reportez-vous [DeleteVpnConnectionRoute](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la route statique spécifiée de la connexion VPN spécifiée. Vous êtes invité à confirmer avant de poursuivre l'opération, sauf si vous spécifiez également le paramètre Force.

```
Remove-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock
11.12.0.0/16
```

Sortie :

```
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"):
```

- Pour plus de détails sur l'API, reportez-vous [DeleteVpnConnectionRoute](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation `DeleteVpnGateway` avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DeleteVpnGateway`.

CLI

AWS CLI

Pour supprimer une passerelle privée virtuelle

Cet exemple supprime la passerelle privée virtuelle spécifiée. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 delete-vpn-gateway --vpn-gateway-id vgw-9a4cacf3
```

- Pour plus de détails sur l'API, reportez-vous [DeleteVpnGateway](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la passerelle privée virtuelle spécifiée. Vous êtes invité à confirmer avant de poursuivre l'opération, sauf si vous spécifiez également le paramètre `Force`.

```
Remove-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

Sortie :

```
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"):
```

- Pour plus de détails sur l'API, reportez-vous [DeleteVpnGateway](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DeregisterImage** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DeregisterImage`.

CLI

AWS CLI

Pour annuler l'enregistrement d'un AMI

Cet exemple désenregistre l'AMI spécifiée. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 deregister-image --image-id ami-4fa54026
```

- Pour plus de détails sur l'API, reportez-vous [DeregisterImage](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple annule l'enregistrement de l'AMI spécifiée.

```
Unregister-EC2Image -ImageId ami-12345678
```

- Pour plus de détails sur l'API, reportez-vous [DeregisterImage](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeAccountAttributes** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeAccountAttributes`.

CLI

AWS CLI

Pour décrire tous les attributs de votre AWS compte

Cet exemple décrit les attributs de votre AWS compte.

Commande :

```
aws ec2 describe-account-attributes
```

Sortie :

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

Pour décrire un attribut unique de votre AWS compte

Cet exemple décrit l'`supported-platforms` attribut de votre AWS compte.

Commande :

```
aws ec2 describe-account-attributes --attribute-names supported-platforms
```

Sortie :

```
{
  "AccountAttributes": [
    {
      "AttributeName": "supported-platforms",
      "AttributeValues": [
        {
          "AttributeValue": "EC2"
        },
        {
          "AttributeValue": "VPC"
        }
      ]
    }
  ]
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeAccountAttributes](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple indique si vous pouvez lancer des instances dans EC2-Classic et EC2-VPC dans la région, ou uniquement dans EC2-VPC.

```
(Get-EC2AccountAttribute -AttributeName supported-platforms).AttributeValues
```

Sortie :

```
AttributeValue
-----
EC2
VPC
```

Exemple 2 : Cet exemple décrit votre VPC par défaut, ou indique « aucun » si vous n'avez pas de VPC par défaut dans la région.

```
(Get-EC2AccountAttribute -AttributeName default-vpc).AttributeValues
```

Sortie :

```
AttributeValue  
-----  
vpc-12345678
```

Exemple 3 : Cet exemple décrit le nombre maximal d'instances à la demande que vous pouvez exécuter.

```
(Get-EC2AccountAttribute -AttributeName max-instances).AttributeValues
```

Sortie :

```
AttributeValue  
-----  
20
```

- Pour plus de détails sur l'API, reportez-vous [DescribeAccountAttributes](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeAddresses** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeAddresses`.

C++

Kit de développement logiciel (SDK) for C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::DescribeAddressesRequest request;
auto outcome = ec2Client.DescribeAddresses(request);
if (outcome.IsSuccess()) {
    std::cout << std::left << std::setw(20) << "InstanceId" <<
        std::setw(15) << "Public IP" << std::setw(10) << "Domain" <<
        std::setw(30) << "Allocation ID" << std::setw(25) <<
        "NIC ID" << std::endl;

    const auto &addresses = outcome.GetResult().GetAddresses();
    for (const auto &address: addresses) {
        Aws::String domainString =
            Aws::EC2::Model::DomainTypeMapper::GetNameForDomainType(
                address.GetDomain());

        std::cout << std::left << std::setw(20) <<
            address.GetInstanceId() << std::setw(15) <<
            address.GetPublicIp() << std::setw(10) << domainString <<
            std::setw(30) << address.GetAllocationId() << std::setw(25)
            << address.GetNetworkInterfaceId() << std::endl;
    }
}
else {
    std::cerr << "Failed to describe Elastic IP addresses:" <<
        outcome.GetError().GetMessage() << std::endl;
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeAddresses](#) à la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Exemple 1 : récupération des détails de toutes vos adresses IP Elastic

L'exemple `describe addresses` suivant affiche les détails de vos adresses IP Elastic.

```
aws ec2 describe-addresses
```

Sortie :

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

Exemple 2 : récupération des détails de vos adresses IP Elastic pour EC2-VPC

L'exemple `describe-addresses` suivant affiche les détails de vos adresses IP Elastic à utiliser avec des instances dans un VPC.

```
aws ec2 describe-addresses \
  --filters "Name=domain,Values=vpc"
```

Sortie :

```
{
  "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"
    }
]
}

```

Exemple 3 : récupération des détails d'une adresse IP Elastic spécifiée selon l'identifiant d'allocation

L'exemple `describe-addresses` suivant affiche des détails sur l'adresse IP Elastic avec l'ID d'allocation spécifié, qui est associée à une instance dans EC2-VPC.

```

aws ec2 describe-addresses \
  --allocation-ids eipalloc-282d9641

```

Sortie :

```

{
  "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"
    }
  ]
}

```

Exemple 4 : récupération des détails d'une adresse IP Elastic spécifiée au moyen de son adresse IP privée de VPC

L'exemple `describe-addresses` suivant affiche des détails sur l'adresse IP Elastic associée à une adresse IP privée particulière dans EC2-VPC.

```

aws ec2 describe-addresses \
  --filters "Name=private-ip-address,Values=10.251.50.12"

```

Exemple 5 : récupération des détails de toutes vos adresses IP Elastic dans EC2-Classic

L'exemple `describe-addresses` suivant affiche des détails sur vos adresses IP Elastic à utiliser dans EC2-Classic.

```
aws ec2 describe-addresses \  
  --filters "Name=domain,Values=standard"
```

Sortie :

```
{  
  "Addresses": [  
    {  
      "InstanceId": "i-1234567890abcdef0",  
      "PublicIp": "203.0.110.25",  
      "PublicIpv4Pool": "amazon",  
      "Domain": "standard"  
    }  
  ]  
}
```

Exemple 6 : récupération des détails d'une adresse IP Elastic spécifiée au moyen de son adresse IP publique

L'exemple `describe-addresses` suivant affiche des détails sur l'adresse IP Elastic avec la valeur `203.0.110.25`, qui est associée à une instance dans EC2-Classic.

```
aws ec2 describe-addresses \  
  --public-ips 203.0.110.25
```

Sortie :

```
{  
  "Addresses": [  
    {  
      "InstanceId": "i-1234567890abcdef0",  
      "PublicIp": "203.0.110.25",  
      "PublicIpv4Pool": "amazon",  
      "Domain": "standard"  
    }  
  ]  
}
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeAddresses](#) à la section Référence des AWS CLI commandes.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

export const main = async () => {
  const command = new DescribeAddressesCommand({
    // You can omit this property to show all addresses.
    AllocationIds: ["ALLOCATION_ID"],
  });

  try {
    const { Addresses } = await client.send(command);
    const addressList = Addresses.map((address) => ` • ${address.PublicIp}`);
    console.log("Elastic IP addresses:");
    console.log(addressList.join("\n"));
  } catch (err) {
    console.error(err);
  }
};
```

- Pour plus de détails sur l'API, reportez-vous [DescribeAddresses](#) à la section Référence des AWS SDK for JavaScript API.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit l'adresse IP élastique spécifiée pour les instances dans EC2-Classic.

```
Get-EC2Address -AllocationId eipalloc-12345678
```

Sortie :

```
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
```

Exemple 2 : Cet exemple décrit vos adresses IP élastiques pour les instances d'un VPC. Cette syntaxe nécessite PowerShell la version 3 ou ultérieure.

```
Get-EC2Address -Filter @{ Name="domain";Values="vpc" }
```

Exemple 3 : Cet exemple décrit l'adresse IP élastique spécifiée pour les instances dans EC2-Classic.

```
Get-EC2Address -PublicIp 203.0.113.17
```

Sortie :

```
AllocationId      :
AssociationId     :
Domain           : standard
InstanceId       : i-12345678
NetworkInterfaceId :
NetworkInterfaceOwnerId :
PrivateIpAddress  :
PublicIp         : 203.0.113.17
```

Exemple 4 : Cet exemple décrit vos adresses IP élastiques pour les instances dans EC2-Classic. Cette syntaxe nécessite PowerShell la version 3 ou ultérieure.

```
Get-EC2Address -Filter @{ Name="domain";Values="standard" }
```

Exemple 5 : Cet exemple décrit toutes vos adresses IP Elastic.

```
Get-EC2Address
```

Exemple 6 : Cet exemple renvoie les adresses IP publique et privée pour l'identifiant d'instance fourni dans le filtre

```
Get-EC2Address -Region eu-west-1 -Filter @{Name="instance-id";Values="i-0c12d3f4f567ffb89"} | Select-Object PrivateIpAddress, PublicIp
```

Sortie :

```
PrivateIpAddress PublicIp
-----
10.0.0.99          63.36.5.227
```

Exemple 7 : Cet exemple récupère toutes les adresses IP élastiques avec leur identifiant d'allocation, leur identifiant d'association et leurs identifiants d'instance

```
Get-EC2Address -Region eu-west-1 | Select-Object InstanceId, AssociationId, AllocationId, PublicIp
```

Sortie :

```
InstanceId          AssociationId        AllocationId
PublicIp
-----
-----
-----
17.212.120.178
i-0c123dfd3415bac67 eipassoc-0e123456bb7890bdb eipalloc-012e3b456789e1fad
17.212.124.77
eipalloc-01cd23ebf45f7890c
17.212.225.7
eipalloc-012345678eeabcfad
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
```

Exemple 8 : Cet exemple extrait la liste des adresses IP EC2 correspondant à la clé de balise « Catégorie » à la valeur « Prod »

```
Get-EC2Address -Filter @{"Name"="tag:Category";Values="Prod"}
```

Sortie :

```
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}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeAddresses](#) à la section Référence des AWS Tools for PowerShell applets de commande.

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
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.
```

- Pour plus de détails sur l'API, reportez-vous [DescribeAddresses](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeAvailabilityZones** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeAvailabilityZones`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Créer et gérer un service résilient](#)

.NET

AWS SDK for .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Get a list of Availability Zones in the AWS Region of the Amazon EC2
Client.
/// </summary>
/// <returns>A list of availability zones.</returns>
public async Task<List<string>> DescribeAvailabilityZones()
{
    var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
        new DescribeAvailabilityZonesRequest());
    return zoneResponse.AvailabilityZones.Select(z => z.ZoneName).ToList();
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeAvailabilityZones](#) à la section Référence des AWS SDK for .NET API.

C++

SDK pour C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

Aws::EC2::Model::DescribeAvailabilityZonesRequest describe_request;
auto describe_outcome =
ec2Client.DescribeAvailabilityZones(describe_request);

if (describe_outcome.IsSuccess()) {
    std::cout << std::left <<
        std::setw(32) << "ZoneName" <<
        std::setw(20) << "State" <<
        std::setw(32) << "Region" << std::endl;

    const auto &zones =
        describe_outcome.GetResult().GetAvailabilityZones();
}
```

```
for (const auto &zone: zones) {
    Aws::String stateString =

    Aws::EC2::Model::AvailabilityZoneStateMapper::GetNameForAvailabilityZoneState(
        zone.GetState());
    std::cout << std::left <<
        std::setw(32) << zone.GetZoneName() <<
        std::setw(20) << stateString <<
        std::setw(32) << zone.GetRegionName() << std::endl;
}
}
else {
    std::cerr << "Failed to describe availability zones:" <<
        describe_outcome.GetError().GetMessage() << std::endl;
    result = false;
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeAvailabilityZones](#) à la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Description de vos zones de disponibilité

L'exemple suivant `describe-availability-zones` affiche les détails des zones de disponibilité mises à votre disposition. La réponse n'inclut que les zones de disponibilité pour la région actuelle. Dans cet exemple, la région par défaut du profil `us-west-2` (Oregon) est utilisée.

```
aws ec2 describe-availability-zones
```

Sortie :

```
{
  "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"
    }
]
}

```

- Pour plus de détails sur l'API, reportez-vous [DescribeAvailabilityZones](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit les zones de disponibilité disponibles pour la région actuelle.

```
Get-EC2AvailabilityZone
```

Sortie :

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

Exemple 2 : Cet exemple décrit toutes les zones de disponibilité dont l'état est altéré. La syntaxe utilisée dans cet exemple nécessite PowerShell la version 3 ou supérieure.

```
Get-EC2AvailabilityZone -Filter @{ Name="state";Values="impaired" }
```

Exemple 3 : Avec PowerShell la version 2, vous devez utiliser New-Object pour créer le filtre.

```

$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = "impaired"

Get-EC2AvailabilityZone -Filter $filter

```

- Pour plus de détails sur l'API, reportez-vous [DescribeAvailabilityZones](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Python

SDK pour Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

    def __init__(
        self,
        resource_prefix,
        inst_type,
        ami_param,
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    ):
        """
        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is
                created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
```

```
self.iam_client = iam_client
self.launch_template_name = f"{resource_prefix}-template"
self.group_name = f"{resource_prefix}-group"
self.instance_policy_name = f"{resource_prefix}-pol"
self.instance_role_name = f"{resource_prefix}-role"
self.instance_profile_name = f"{resource_prefix}-prof"
self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
self.bad_creds_role_name = f"{resource_prefix}-bc-role"
self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"
self.key_pair_name = f"{resource_prefix}-key-pair"

def get_availability_zones(self):
    """
    Gets a list of Availability Zones in the AWS Region of the Amazon EC2
    client.

    :return: The list of Availability Zones for the client Region.
    """
    try:
        response = self.ec2_client.describe_availability_zones()
        zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
    except ClientError as err:
        raise AutoScalerError(f"Couldn't get availability zones: {err}.")
    else:
        return zones
```

- Pour plus de détails sur l'API, consultez [DescribeAvailabilityZones](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
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.
```

- Pour plus de détails sur l'API, reportez-vous [DescribeAvailabilityZones](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeBundleTasks** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeBundleTasks`.

CLI

AWS CLI

Pour décrire les tâches de votre bundle

Cet exemple décrit toutes vos tâches groupées.

Commande :

```
aws ec2 describe-bundle-tasks
```

Sortie :

```
{
```

```
"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"  
  }  
]
```

- Pour plus de détails sur l'API, reportez-vous [DescribeBundleTasks](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit la tâche groupée spécifiée.

```
Get-EC2BundleTask -BundleId bun-12345678
```

Exemple 2 : Cet exemple décrit les tâches groupées dont l'état est « terminé » ou « échec ».

```
$filter = New-Object Amazon.EC2.Model.Filter  
$filter.Name = "state"  
$filter.Values = @( "complete", "failed" )  
  
Get-EC2BundleTask -Filter $filter
```

- Pour plus de détails sur l'API, reportez-vous [DescribeBundleTasks](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeCapacityReservations** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeCapacityReservations`.

CLI

AWS CLI

Exemple 1 : Pour décrire une ou plusieurs de vos réservations de capacité

L'`describe-capacity-reservation` exemple suivant affiche les détails de toutes vos réservations de capacité dans la AWS région actuelle.

```
aws ec2 describe-capacity-reservations
```

Sortie :

```
{
  "CapacityReservations": [
    {
      "CapacityReservationId": "cr-1234abcd56EXAMPLE ",
      "EndDateType": "unlimited",
      "AvailabilityZone": "eu-west-1a",
      "InstanceMatchCriteria": "open",
      "Tags": [],
      "EphemeralStorage": false,
      "CreateDate": "2019-08-16T09:03:18.000Z",
      "AvailableInstanceCount": 1,
      "InstancePlatform": "Linux/UNIX",
      "TotalInstanceCount": 1,
      "State": "active",
      "Tenancy": "default",
      "EbsOptimized": true,
      "InstanceType": "a1.medium"
    },
    {
```

```

    "CapacityReservationId": "cr-abcdEXAMPLE9876ef ",
    "EndDateType": "unlimited",
    "AvailabilityZone": "eu-west-1a",
    "InstanceMatchCriteria": "open",
    "Tags": [],
    "EphemeralStorage": false,
    "CreateDate": "2019-08-07T11:34:19.000Z",
    "AvailableInstanceCount": 3,
    "InstancePlatform": "Linux/UNIX",
    "TotalInstanceCount": 3,
    "State": "cancelled",
    "Tenancy": "default",
    "EbsOptimized": true,
    "InstanceType": "m5.large"
  }
]
}

```

Exemple 2 : Pour décrire une ou plusieurs de vos réservations de capacité

L'`describe-capacity-reservationsexemple` suivant affiche les détails relatifs à la réservation de capacité spécifiée.

```

aws ec2 describe-capacity-reservations \
  --capacity-reservation-ids cr-1234abcd56EXAMPLE

```

Sortie :

```

{
  "CapacityReservations": [
    {
      "CapacityReservationId": "cr-1234abcd56EXAMPLE",
      "EndDateType": "unlimited",
      "AvailabilityZone": "eu-west-1a",
      "InstanceMatchCriteria": "open",
      "Tags": [],
      "EphemeralStorage": false,
      "CreateDate": "2019-08-16T09:03:18.000Z",
      "AvailableInstanceCount": 1,
      "InstancePlatform": "Linux/UNIX",
      "TotalInstanceCount": 1,
      "State": "active",
      "Tenancy": "default",
    }
  ]
}

```

```
        "EbsOptimized": true,  
        "InstanceType": "a1.medium"  
    }  
]  
}
```

Pour plus d'informations, consultez la section [Visualisation d'une réservation de capacité](#) dans le guide de l'utilisateur Amazon Elastic Compute Cloud pour les instances Linux.

- Pour plus de détails sur l'API, reportez-vous [DescribeCapacityReservations](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit une ou plusieurs de vos réservations de capacité pour la région

```
Get-EC2CapacityReservation -Region eu-west-1
```

Sortie :

```
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
```

- Pour plus de détails sur l'API, reportez-vous [DescribeCapacityReservations](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation `DescribeCustomerGateways` avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeCustomerGateways`.

CLI

AWS CLI

Pour décrire les passerelles de vos clients

Cet exemple décrit les passerelles de vos clients.

Commande :

```
aws ec2 describe-customer-gateways
```

Sortie :

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

Pour décrire une passerelle client spécifique

Cet exemple décrit la passerelle client spécifiée.

Commande :

```
aws ec2 describe-customer-gateways --customer-gateway-ids cgw-0e11f167
```

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [DescribeCustomerGateways](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit la passerelle client spécifiée.

```
Get-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

Sortie :

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

Exemple 2 : Cet exemple décrit toute passerelle client dont l'état est en attente ou disponible.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2CustomerGateway -Filter $filter
```

Exemple 3 : Cet exemple décrit toutes vos passerelles clients.

```
Get-EC2CustomerGateway
```

- Pour plus de détails sur l'API, reportez-vous [DescribeCustomerGateways](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeDhcpOptions** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeDhcpOptions`.

CLI

AWS CLI

Exemple 1 : Pour décrire vos options DHCP

L'`describe-dhcp-optionsexemple` suivant permet de récupérer des informations sur vos options DHCP.

```
aws ec2 describe-dhcp-options
```

Sortie :

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

Pour plus d'informations, consultez la section [Utilisation des ensembles d'options DHCP](#) dans le guide de l'AWS utilisateur VPC.

Exemple 2 : pour décrire vos options DHCP et filtrer la sortie

L'`describe-dhcp-options` exemple suivant décrit vos options DHCP et utilise un filtre pour renvoyer uniquement les options DHCP correspondant au serveur `example.com` de noms de domaine. L'exemple utilise le `--query` paramètre pour afficher uniquement les informations de configuration et l'ID dans la sortie.

```
aws ec2 describe-dhcp-options \
  --filters Name=key,Values=domain-name-servers Name=value,Values=example.com \
  --query "DhcpOptions[*].[DhcpConfigurations,DhcpOptionsId]"
```

Sortie :

```
[
  [
    [
      {
        "Key": "domain-name",
        "Values": [
          {
            "Value": "example.com"
          }
        ]
      },
      {
        "Key": "domain-name-servers",
        "Values": [
          {
            "Value": "172.16.16.16"
          }
        ]
      }
    ],
    "dopt-001122334455667ab"
  ]
]
```

Pour plus d'informations, consultez la section [Utilisation des ensembles d'options DHCP](#) dans le guide de l'AWS utilisateur VPC.

- Pour plus de détails sur l'API, reportez-vous [DescribeDhcpOptions](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie vos ensembles d'options DHCP.

```
Get-EC2DhcpOption
```

Sortie :

DhcpConfigurations	DhcpOptionsId	Tag
-----	-----	---
{domain-name, domain-name-servers}	dopt-1a2b3c4d	{}
{domain-name, domain-name-servers}	dopt-2a3b4c5d	{}
{domain-name-servers}	dopt-3a4b5c6d	{}

Exemple 2 : Cet exemple permet d'obtenir les détails de configuration du jeu d'options DHCP spécifié.

```
(Get-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d).DhcpConfigurations
```

Sortie :

Key	Values
---	-----
domain-name	{abc.local}
domain-name-servers	{10.0.0.101, 10.0.0.102}

- Pour plus de détails sur l'API, reportez-vous [DescribeDhcpOptions](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeFlowLogs** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeFlowLogs`.

CLI

AWS CLI

Exemple 1 : pour décrire tous vos journaux de flux

L'`describe-flow-log`exemple suivant affiche les détails de tous vos journaux de flux.

```
aws ec2 describe-flow-logs
```

Sortie :

```
{
  "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}"
    }
]
}

```

Exemple 2 : pour décrire un sous-ensemble de vos journaux de flux

L'`describe-flow-log` exemple suivant utilise un filtre pour afficher les détails uniquement pour les journaux de flux qui se trouvent dans le groupe de CloudWatch journaux spécifié dans Amazon Logs.

```

aws ec2 describe-flow-logs \
    --filter "Name=log-group-name,Values=MyFlowLogs"

```

- Pour plus de détails sur l'API, reportez-vous [DescribeFlowLogs](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit un ou plusieurs journaux de flux avec le type de destination de journal « s3 »

```

Get-EC2FlowLog -Filter @{Name="log-destination-type";Values="s3"}

```

Sortie :

```

CreationTime           : 2/25/2019 9:07:36 PM
DeliverLogsErrorMessage :
DeliverLogsPermissionArn :
DeliverLogsStatus      : SUCCESS
FlowLogId              : f1-01b2e3d45f67f8901
FlowLogStatus          : ACTIVE
LogDestination         : arn:aws:s3::my-bucket-dd-tata
LogDestinationType     : s3
LogGroupName          :

```

```
ResourceId           : eni-01d2dda3456b7e890
TrafficType          : ALL
```

- Pour plus de détails sur l'API, reportez-vous [DescribeFlowLogs](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeHostReservationOfferings** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeHostReservationOfferings`.

CLI

AWS CLI

Pour décrire les offres de réservation d'hôtes dédiés

Cet exemple décrit les réservations d'hôtes dédiés pour la famille d'instances M4 disponibles à l'achat.

Commande :

```
aws ec2 describe-host-reservation-offerings --filter Name=instance-family,Values=m4
```

Sortie :

```
{
  "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
    }
  ]
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeHostReservationOfferings](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit les réservations d'hôtes dédiés disponibles à l'achat pour le filtre « instance-family » donné où se PaymentOption trouve « » NoUpfront

```
Get-EC2HostReservationOffering -Filter @{Name="instance-family";Values="m4"} |  
Where-Object PaymentOption -eq NoUpfront
```

Sortie :

```
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
```

- Pour plus de détails sur l'API, reportez-vous [DescribeHostReservationOfferings](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeHosts** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeHosts`.

CLI

AWS CLI

Pour afficher des informations sur les hôtes dédiés

L'exemple suivant affiche les détails des hôtes disponibles dédiés de votre compte AWS.

```
aws ec2 describe-hosts --filter "Name=state,Values=available"
```

Sortie :

```
{
  "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"
}
]
```

Pour plus d'informations, consultez la section [Visualisation des hôtes dédiés](#) dans le guide de l'utilisateur Amazon Elastic Compute Cloud pour les instances Linux.

- Pour plus de détails sur l'API, reportez-vous [DescribeHosts](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : cet exemple renvoie les détails de l'hôte EC2

```
Get-EC2Host
```

Sortie :

```
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                : {}
```

Exemple 2 : Cet exemple interroge le `AvailableInstanceCapacity` pour l'hôte `h-01e23f4cd567899f1`

```
Get-EC2Host -HostId h-01e23f4cd567899f1 | Select-Object -ExpandProperty
AvailableCapacity | Select-Object -expand AvailableInstanceCapacity
```

Sortie :

```
AvailableCapacity InstanceType TotalCapacity
-----
11                m4.xlarge    11
```

- Pour plus de détails sur l'API, reportez-vous [DescribeHosts](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeIamInstanceProfileAssociations** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeIamInstanceProfileAssociations`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Créer et gérer un service résilient](#)

.NET

AWS SDK for .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Get the instance profile association data for an instance.
/// </summary>
/// <param name="instanceId">The Id of the instance.</param>
/// <returns>Instance profile associations data.</returns>
public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
{
    var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
    new DescribeIamInstanceProfileAssociationsRequest()
    {
        Filters = new List<Amazon.EC2.Model.Filter>()
        {
            new ("instance-id", new List<string>() { instanceId })
        },
    });
    return response.IamInstanceProfileAssociations[0];
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeIamInstanceProfileAssociations](#) à la section Référence des AWS SDK for .NET API.

CLI

AWS CLI

Description des associations de profils d'instances IAM

Cet exemple décrit toutes vos associations de profils d'instance IAM.

Commande :

```
aws ec2 describe-iam-instance-profile-associations
```

Sortie :

```
{
  "IamInstanceProfileAssociations": [
    {
```

```

    "InstanceId": "i-09eb09efa73ec1dee",
    "State": "associated",
    "AssociationId": "iip-assoc-0db249b1f25fa24b8",
    "IamInstanceProfile": {
      "Id": "AIPAJVQN4F5WVLGCJDRGM",
      "Arn": "arn:aws:iam::123456789012:instance-profile/admin-role"
    }
  },
  {
    "InstanceId": "i-0402909a2f4dff14",
    "State": "associating",
    "AssociationId": "iip-assoc-0d1ec06278d29f44a",
    "IamInstanceProfile": {
      "Id": "AGJAJVQN4F5WVLGCJABCM",
      "Arn": "arn:aws:iam::123456789012:instance-profile/user1-role"
    }
  }
]
}

```

- Pour plus de détails sur l'API, reportez-vous [DescribeIamInstanceProfileAssociations](#) à la section Référence des AWS CLI commandes.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

const ec2Client = new EC2Client({});
const { IamInstanceProfileAssociations } = await ec2Client.send(
  new DescribeIamInstanceProfileAssociationsCommand({
    Filters: [
      { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
    ],
  }),
);

```

- Pour plus de détails sur l'API, reportez-vous [DescribeIamInstanceProfileAssociations](#) à la section Référence des AWS SDK for JavaScript API.

Python

SDK pour Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

    def __init__(
        self,
        resource_prefix,
        inst_type,
        ami_param,
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    ):
        """
        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is
            created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
```

```

"""
self.inst_type = inst_type
self.ami_param = ami_param
self.autoscaling_client = autoscaling_client
self.ec2_client = ec2_client
self.ssm_client = ssm_client
self.iam_client = iam_client
self.launch_template_name = f"{resource_prefix}-template"
self.group_name = f"{resource_prefix}-group"
self.instance_policy_name = f"{resource_prefix}-pol"
self.instance_role_name = f"{resource_prefix}-role"
self.instance_profile_name = f"{resource_prefix}-prof"
self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
self.bad_creds_role_name = f"{resource_prefix}-bc-role"
self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"
self.key_pair_name = f"{resource_prefix}-key-pair"

def get_instance_profile(self, instance_id):
    """
    Gets data about the profile associated with an instance.

    :param instance_id: The ID of the instance to look up.
    :return: The profile data.
    """
    try:
        response =
self.ec2_client.describe_iam_instance_profile_associations(
            Filters=[{"Name": "instance-id", "Values": [instance_id]}]
        )
    except ClientError as err:
        raise AutoScalerError(
            f"Couldn't get instance profile association for instance
{instance_id}: {err}"
        )
    else:
        return response["IamInstanceProfileAssociations"][0]

```

- Pour plus de détails sur l'API, consultez [DescribeIamInstanceProfileAssociations](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeIdFormat** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeIdFormat`.

CLI

AWS CLI

Exemple 1 : pour décrire le format d'identification d'une ressource

L'`describe-id-format` exemple suivant décrit le format d'identification des groupes de sécurité.

```
aws ec2 describe-id-format \  
  --resource security-group
```

Dans l'exemple de sortie suivant, la `Deadline` valeur indique que la date limite pour que ce type de ressource passe définitivement du format d'identifiant court au format d'identifiant long a expiré à 00h00 UTC le 15 août 2018.

```
{  
  "Statuses": [  
    {  
      "Deadline": "2018-08-15T00:00:00.000Z",  
      "Resource": "security-group",  
      "UseLongIds": true  
    }  
  ]  
}
```

Exemple 2 : pour décrire le format d'identification de toutes les ressources

L'`describe-id-format` exemple suivant décrit le format d'ID pour tous les types de ressources. Tous les types de ressources prenant en charge le format d'identifiant court ont été changés pour utiliser le format d'identifiant long.

```
aws ec2 describe-id-format
```

- Pour plus de détails sur l'API, reportez-vous [DescribeIdFormat](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit le format d'ID pour le type de ressource spécifié.

```
Get-EC2IdFormat -Resource instance
```

Sortie :

```
Resource      UseLongIds
-----
instance      False
```

Exemple 2 : Cet exemple décrit les formats d'identification pour tous les types de ressources qui prennent en charge les identifiants plus longs.

```
Get-EC2IdFormat
```

Sortie :

```
Resource      UseLongIds
-----
reservation   False
instance      False
```

- Pour plus de détails sur l'API, reportez-vous [DescribeIdFormat](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeIdentityIdFormat** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeIdentityIdFormat`.

CLI

AWS CLI

Pour décrire le format d'ID d'un rôle IAM

L'`describe-identity-id-format` exemple suivant décrit le format d'ID reçu par les instances créées par le rôle IAM `EC2Role` dans votre AWS compte.

```
aws ec2 describe-identity-id-format \  
  --principal-arn arn:aws:iam::123456789012:role/my-iam-role \  
  --resource instance
```

Le résultat suivant indique que les instances créées par ce rôle reçoivent des identifiants au format ID long.

```
{  
  "Statuses": [  
    {  
      "Deadline": "2016-12-15T00:00:00Z",  
      "Resource": "instance",  
      "UseLongIds": true  
    }  
  ]  
}
```

Pour décrire le format d'identification d'un utilisateur IAM

L'`describe-identity-id-format` exemple suivant décrit le format d'identification reçu par les instantanés créés par l'utilisateur IAM `AdminUser` dans votre AWS compte.

```
aws ec2 describe-identity-id-format \  
  --principal-arn arn:aws:iam::123456789012:user/AdminUser \  
  --resource snapshot
```

La sortie indique que les instantanés créés par cet utilisateur reçoivent des identifiants au format ID long.

```
{  
  "Statuses": [  

```

```

    {
      "Deadline": "2016-12-15T00:00:00Z",
      "Resource": "snapshot",
      "UseLongIds": true
    }
  ]
}

```

- Pour plus de détails sur l'API, reportez-vous [DescribeIdentityIdFormat](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie le format d'identifiant de la ressource « image » pour le rôle donné

```

Get-EC2IdentityIdFormat -PrincipalArn arn:aws:iam::123456789511:role/JDBC -
Resource image

```

Sortie :

```

Deadline           Resource UseLongIds
-----
8/2/2018 11:30:00 PM image      True

```

- Pour plus de détails sur l'API, reportez-vous [DescribeIdentityIdFormat](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeImageAttribute** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeImageAttribute`.

CLI

AWS CLI

Pour décrire les autorisations de lancement d'une AMI

Cet exemple décrit les autorisations de lancement pour l'AMI spécifiée.

Commande :

```
aws ec2 describe-image-attribute --image-id ami-5731123e --attribute
  launchPermission
```

Sortie :

```
{
  "LaunchPermissions": [
    {
      "UserId": "123456789012"
    }
  ],
  "ImageId": "ami-5731123e",
}
```

Pour décrire les codes de produit pour une AMI

Cet exemple décrit les codes de produit pour l'AMI spécifiée. Notez que cette AMI ne contient aucun code produit.

Commande :

```
aws ec2 describe-image-attribute --image-id ami-5731123e --attribute productCodes
```

Sortie :

```
{
  "ProductCodes": [],
  "ImageId": "ami-5731123e",
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeImageAttribute](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple obtient la description de l'AMI spécifiée.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute description
```

Sortie :

```
BlockDeviceMappings : {}  
Description          : My image description  
ImageId              : ami-12345678  
KernelId             :  
LaunchPermissions    : {}  
ProductCodes         : {}  
RamdiskId            :  
SriovNetSupport      :
```

Exemple 2 : Cet exemple obtient les autorisations de lancement pour l'AMI spécifiée.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

Sortie :

```
BlockDeviceMappings : {}  
Description          :  
ImageId              : ami-12345678  
KernelId             :  
LaunchPermissions    : {all}  
ProductCodes         : {}  
RamdiskId            :  
SriovNetSupport      :
```

Exemple 3 : Cet exemple teste si la mise en réseau améliorée est activée.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute sriovNetSupport
```

Sortie :

```
BlockDeviceMappings : {}
```

```
Description      :
ImageId          : ami-12345678
KernelId        :
LaunchPermissions : {}
ProductCodes     : {}
RamdiskId       :
SriovNetSupport  : simple
```

- Pour plus de détails sur l'API, reportez-vous [DescribeImageAttribute](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeImages** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeImages`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Démarrer avec les instances](#)

Bash

AWS CLI avec le script Bash

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
#####
# 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
}

```

Fonctions utilitaires utilisées dans cet exemple.

```

#####
# 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
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeImages](#) à la section Référence des AWS CLI commandes.

CLI

AWS CLI

Exemple 1 : description d'une AMI

L'exemple `describe-images` suivant décrit l'AMI spécifiée dans la région spécifiée.

```
aws ec2 describe-images \  
  --region us-east-1 \  
  --image-ids ami-1234567890EXAMPLE
```

Sortie :

```
{  
  "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"
  }
]
}

```

Pour plus d'informations, veuillez consulter la rubrique [Amazon Machine Images \(AMI\)](#) dans le Guide de l'utilisateur Amazon EC2.

Exemple 2 : description d'AMI sur la base de filtres

L'exemple `describe-images` suivant décrit les AMI Windows fournies par Amazon qui sont sauvegardées par Amazon EBS.

```
aws ec2 describe-images \
```

```
--owners amazon \  
--filters "Name=platform,Values=windows" "Name=root-device-type,Values=efs"
```

Pour obtenir un exemple de la manière de procéder pour `describe-images`, reportez-vous à l'exemple 1.

Pour d'autres exemples d'utilisation de filtres, veuillez consulter la rubrique [Lister et filtrer vos ressources](#) dans le Guide de l'utilisateur Amazon EC2.

Exemple 3 : description d'AMI sur la base de balises

L'exemple `describe-images` suivant décrit toutes les AMI comportant la balise `Type=Custom`. L'exemple utilise le paramètre `--query` pour n'afficher que les ID des AMI.

```
aws ec2 describe-images \  
  --filters "Name=tag:Type,Values=Custom" \  
  --query 'Images[*].[ImageId]' \  
  --output text
```

Sortie :

```
ami-1234567890EXAMPLE  
ami-0abcdef1234567890
```

Pour d'autres exemples d'utilisation de filtres de balises, veuillez consulter la rubrique [Utiliser des balises](#) dans le Guide de l'utilisateur Amazon EC2.

- Pour plus de détails sur l'API, reportez-vous [DescribeImages](#) à la section Référence des AWS CLI commandes.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

// List at least the first i386 image available for EC2 instances.
export const main = async () => {
  // The paginate function is a wrapper around the base command.
  const paginator = paginateDescribeImages(
    // Without limiting the page size, this call can take a long time. pageSize
    // is just sugar for
    // the MaxResults property in the base command.
    { client, pageSize: 25 },
    {
      // There are almost 70,000 images available. Be specific with your
      // filtering
      // to increase efficiency.
      // See https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/clients/
      // client-ec2/interfaces/describeimagescommandinput.html#filters
      Filters: [{ Name: "architecture", Values: ["x86_64"] }],
    },
  );

  try {
    const arm64Images = [];
    for await (const page of paginator) {
      if (page.Images.length) {
        arm64Images.push(...page.Images);
        // Once we have at least 1 result, we can stop.
        if (arm64Images.length >= 1) {
          break;
        }
      }
    }
    console.log(arm64Images);
  } catch (err) {
    console.error(err);
  }
};
```

- Pour plus de détails sur l'API, reportez-vous [DescribeImages](#) à la section Référence des AWS SDK for JavaScript API.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit l'AMI spécifiée.

```
Get-EC2Image -ImageId ami-12345678
```

Sortie :

```
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
```

Exemple 2 : Cet exemple décrit les AMI que vous possédez.

```
Get-EC2Image -owner self
```

Exemple 3 : Cet exemple décrit les AMI publiques qui exécutent Microsoft Windows Server.

```
Get-EC2Image -Filter @{ Name="platform"; Values="windows" }
```

Exemple 4 : Cet exemple décrit toutes les AMI publiques de la région « us-west-2 ».

```
Get-EC2Image -Region us-west-2
```

- Pour plus de détails sur l'API, reportez-vous [DescribeImages](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Python

SDK pour Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance
    actions."""

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

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

    def get_images(self, image_ids):
```

```
"""
Gets information about Amazon Machine Images (AMIs) from a list of AMI
IDs.

:param image_ids: The list of AMIs to look up.
:return: A list of Boto3 Image objects that represent the requested AMIs.
"""
try:
    images = list(self.ec2_resource.images.filter(ImageIds=image_ids))
except ClientError as err:
    logger.error(
        "Couldn't get images. Here's why: %s: %s",
        err.response["Error"]["Code"],
        err.response["Error"]["Message"],
    )
    raise
else:
    return images
```

- Pour plus de détails sur l'API, consultez [DescribeImages](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Rust

SDK pour Rust

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// A simple Rust program to list all Amazon images in the currently configured
region.
#[tokio::main]
async fn main() -> Result<(), Error> {
    let config = aws_config::load_from_env().await;
    let client = Client::new(&config);
```

```
let resp = client.describe_images().owners("amazon").send().await?;

println!("AWS SDK for Rust v{}", PKG_VERSION);
println!("Describing Amazon Machine Images (AMIs):");

let mut images: Vec<_> = resp
    .images()
    .iter()
    .filter(|i| {
        i.description()
            .filter(|i| i.contains("Amazon Linux AMI 2023"))
            .is_some()
    })
    .collect();
images.sort_by(|a, b| a.description.cmp(&b.description));

if images.is_empty() {
    println!("No images found.");
    return Ok(());
}

for image in images {
    let id = image.image_id().unwrap_or_default();
    let description = image.description().unwrap_or_default();

    println!("{id}: {description}");
}

Ok(())
}
```

- Pour plus de détails sur l'API, voir [DescribeImages](#) la section de référence de l'API AWS SDK for Rust.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeImportImageTasks** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeImportImageTasks`.

CLI

AWS CLI

Pour surveiller une tâche d'importation d'image

L'`describe-import-image-tasks` suivant vérifie le statut de la tâche d'importation d'image spécifiée.

```
aws ec2 describe-import-image-tasks \
  --import-task-ids import-ami-1234567890abcdef0
```

Sortie pour une tâche d'importation d'image en cours.

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

Sortie pour une tâche d'importation d'image terminée. L'ID de l'AMI qui en résulte est fourni par `ImageId`.

```
{
  "ImportImageTasks": [
    {
      "ImportTaskId": "import-ami-1234567890abcdef0",
```

```
    "ImageId": "ami-1234567890abcdef0",
    "SnapshotDetails": [
      {
        "DiskImageSize": 705638400.0,
        "Format": "ova",
        "SnapshotId": "snap-1234567890abcdef0"
        "Status": "completed",
        "UserBucket": {
          "S3Bucket": "my-import-bucket",
          "S3Key": "vms/my-server-vm.ova"
        }
      }
    ],
    "Status": "completed"
  }
]
```

- Pour plus de détails sur l'API, reportez-vous [DescribeImportImageTasks](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit la tâche d'importation d'image spécifiée.

```
Get-EC2ImportImageTask -ImportTaskId import-ami-hgfedcba
```

Sortie :

```
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     :
```

Exemple 2 : Cet exemple décrit toutes vos tâches d'importation d'images.

```
Get-EC2ImportImageTask
```

Sortie :

```
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     :
```

- Pour plus de détails sur l'API, reportez-vous [DescribeImportImageTasks](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeImportSnapshotTasks** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeImportSnapshotTasks`.

CLI

AWS CLI

Pour surveiller une tâche d'importation de snapshots

L'`describe-import-snapshot-task` suivant vérifie le statut de la tâche de capture instantanée d'importation spécifiée.

```
aws ec2 describe-import-snapshot-tasks \
  --import-task-ids import-snap-1234567890abcdef0
```

Résultat pour une tâche d'importation de snapshots en cours :

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

Sortie pour une tâche d'importation d'instantanés terminée. L'ID de l'instantané obtenu est fourni par `SnapshotId`.

```
{
  "ImportSnapshotTasks": [
    {
```

```

    "Description": "My server VMDK",
    "ImportTaskId": "import-snap-1234567890abcdef0",
    "SnapshotTaskDetail": {
      "Description": "My server VMDK",
      "DiskImageSize": "705638400.0",
      "Format": "VMDK",
      "SnapshotId": "snap-1234567890abcdef0"
      "Status": "completed",
      "UserBucket": {
        "S3Bucket": "my-import-bucket",
        "S3Key": "vms/my-server-vm.vmdk"
      }
    }
  }
]
}

```

- Pour plus de détails sur l'API, reportez-vous [DescribeImportSnapshotTasks](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit la tâche d'importation de clichés spécifiée.

```
Get-EC2ImportSnapshotTask -ImportTaskId import-snap-abcdefgh
```

Sortie :

Description	ImportTaskId	SnapshotTaskDetail
-----	-----	-----
Disk Image Import 1	import-snap-abcdefgh	Amazon.EC2.Model.SnapshotTaskDetail

Exemple 2 : Cet exemple décrit toutes vos tâches d'importation d'instantanés.

```
Get-EC2ImportSnapshotTask
```

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [DescribeImportSnapshotTasks](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation `DescribeInstanceAttribute` avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeInstanceAttribute`.

CLI**AWS CLI**

Pour décrire le type d'instance

Cet exemple décrit le type d'instance de l'instance spécifiée.

Commande :

```
aws ec2 describe-instance-attribute --instance-id i-1234567890abcdef0 --attribute instanceType
```

Sortie :

```
{
  "InstanceId": "i-1234567890abcdef0"
  "InstanceType": {
```

```
    "Value": "t1.micro"
  }
}
```

Pour décrire l' `disableApiTermination` attribut

Cet exemple décrit l'`disableApiTermination` attribut de l'instance spécifiée.

Commande :

```
aws ec2 describe-instance-attribute --instance-id i-1234567890abcdef0 --attribute
disableApiTermination
```

Sortie :

```
{
  "InstanceId": "i-1234567890abcdef0"
  "DisableApiTermination": {
    "Value": "false"
  }
}
```

Pour décrire le mappage des périphériques en mode bloc pour une instance

Cet exemple décrit l'`blockDeviceMapping` attribut de l'instance spécifiée.

Commande :

```
aws ec2 describe-instance-attribute --instance-id i-1234567890abcdef0 --attribute
blockDeviceMapping
```

Sortie :

```
{
  "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"
    }
}
],
}

```

- Pour plus de détails sur l'API, reportez-vous [DescribeInstanceAttribute](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit le type d'instance de l'instance spécifiée.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute instanceType
```

Sortie :

```
InstanceType           : t2.micro
```

Exemple 2 : Cet exemple indique si la mise en réseau améliorée est activée pour l'instance spécifiée.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

Sortie :

```
SriovNetSupport        : simple
```

Exemple 3 : Cet exemple décrit les groupes de sécurité pour l'instance spécifiée.

```
(Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute groupSet).Groups
```

Sortie :

```
GroupId
-----
sg-12345678
sg-45678901
```

Exemple 4 : Cet exemple indique si l'optimisation EBS est activée pour l'instance spécifiée.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

Sortie :

```
EbsOptimized           : False
```

Exemple 5 : Cet exemple décrit l'attribut `disableApiTermination` « » de l'instance spécifiée.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

Sortie :

```
DisableApiTermination   : False
```

Exemple 6 : Cet exemple décrit l'attribut « `instanceInitiatedShutdown Comportement` » de l'instance spécifiée.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
instanceInitiatedShutdownBehavior
```

Sortie :

```
InstanceInitiatedShutdownBehavior : stop
```

- Pour plus de détails sur l'API, reportez-vous [DescribeInstanceAttribute](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeInstanceStatus** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeInstanceStatus`.

CLI

AWS CLI

Description du statut d'une instance

L'exemple `describe-instance-status` suivant décrit le statut actuel de l'instance spécifiée.

```
aws ec2 describe-instance-status \  
  --instance-ids i-1234567890abcdef0
```

Sortie :

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

Pour plus d'informations, veuillez consulter la rubrique [Surveiller le statut de vos instances](#) dans le guide de l'utilisateur Amazon EC2.

- Pour plus de détails sur l'API, reportez-vous [DescribeInstanceStatus](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit le statut de l'instance spécifiée.

```
Get-EC2InstanceStatus -InstanceId i-12345678
```

Sortie :

```
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
```

Sortie :

Code	Name
----	----
16	running

```
$status.Status
```

Sortie :

Details	Status
-----	-----
{reachability}	ok

```
$status.SystemStatus
```

Sortie :

Details	Status
-----	-----
{reachability}	ok

- Pour plus de détails sur l'API, reportez-vous [DescribeInstanceStatus](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Rust**SDK pour Rust****Note**

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
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(())
}

```

- Pour plus de détails sur l'API, voir [DescribeInstanceStatus](#) la section de référence de l'API AWS SDK for Rust.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeInstanceTypes** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeInstanceTypes`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Démarrer avec les instances](#)

.NET

AWS SDK for .NET

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

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

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

- Pour plus de détails sur l'API, reportez-vous [DescribeInstanceTypes](#) à la section Référence des AWS SDK for .NET API.

Bash

AWS CLI avec le script Bash

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
#####
# 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
}

```

Fonctions utilitaires utilisées dans cet exemple.

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

- Pour plus de détails sur l'API, reportez-vous [DescribeInstanceTypes](#) à la section Référence des AWS CLI commandes.

CLI

AWS CLI

Exemple 1 : description d'un type d'instance

L'exemple `describe-instance-types` suivant affiche les détails du type d'instance spécifié.

```
aws ec2 describe-instance-types \  
  --instance-types t2.micro
```

Sortie :

```
{  
  "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
    }
}
```

Pour plus d'informations, consultez les [types d'instances](#) dans le guide de l'utilisateur Amazon Elastic Compute Cloud pour les instances Linux.

Exemple 2 : filtrage des types d'instance disponibles

Vous pouvez définir un filtre pour limiter les résultats aux types d'instances présentant une caractéristique spécifique. L'exemple `describe-instance-types` suivant liste les types d'instance qui prennent en charge l'hibernation.

```
aws ec2 describe-instance-types \
  --filters Name=hibernation-supported,Values=true --query
  'InstanceTypes[*].InstanceType'
```

Sortie :

```
[
  "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"
]
```

Pour plus d'informations, consultez les [types d'instances](#) dans le guide de l'utilisateur Amazon Elastic Compute Cloud pour les instances Linux.

- Pour plus de détails sur l'API, reportez-vous [DescribeInstanceTypes](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Get a list of instance types.
public static String getInstanceTypes(Ec2Client ec2) {
    String instanceType;
    try {
        DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
            .maxResults(10)
            .build();

        DescribeInstanceTypesResponse response =
ec2.describeInstanceTypes(typesRequest);
        List<InstanceTypeInfo> instanceTypes = response.instanceTypes();
        for (InstanceTypeInfo type : instanceTypes) {
            System.out.println("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
            System.out.println("Network information is " +
type.networkInfo().toString());
            System.out.println("Instance type is " +
type.instanceType().toString());
            instanceType = type.instanceType().toString();
            if (instanceType.compareTo("t2.2xlarge") == 0){
                return instanceType;
            }
        }

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeInstanceTypes](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import {
  paginateDescribeInstanceTypes,
  DescribeInstanceTypesCommand,
} from "@aws-sdk/client-ec2";

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

// List at least the first arm64 EC2 instance type available.
export const main = async () => {
  // The paginate function is a wrapper around the underlying command.
  const paginator = paginateDescribeInstanceTypes(
    // Without limiting the page size, this call can take a long time. pageSize
    // is just sugar for
    // the MaxResults property in the underlying command.
    { client, pageSize: 25 },
    {
      Filters: [
        { Name: "processor-info.supported-architecture", Values: ["x86_64"] },
        { Name: "free-tier-eligible", Values: ["true"] },
      ],
    },
  );

  try {
    const instanceTypes = [];

    for await (const page of paginator) {
      if (page.InstanceTypes.length) {
```

```
instanceTypes.push(...page.InstanceTypes);

// When we have at least 1 result, we can stop.
if (instanceTypes.length >= 1) {
    break;
}
}
}
console.log(instanceTypes);
} catch (err) {
    console.error(err);
}
};
```

- Pour plus de détails sur l'API, reportez-vous [DescribeInstanceTypes](#) à la section Référence des AWS SDK for JavaScript API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Get a list of instance types.
suspend fun getInstanceTypesSc(): String {
    var instanceType = ""
    val filterObs = ArrayList<Filter>()
    val filter =
        Filter {
            name = "processor-info.supported-architecture"
            values = listOf("arm64")
        }

    filterObs.add(filter)
    val typesRequest =
        DescribeInstanceTypesRequest {
```

```

        filters = filterObs
        maxResults = 10
    }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeInstanceTypes(typesRequest)
        response.instanceTypes?.forEach { type ->
            println("The memory information of this type is
${type.memoryInfo?.sizeInMib}")
            println("Maximum number of network cards is
${type.networkInfo?.maximumNetworkCards}")
            instanceType = type.instanceType.toString()
        }
        return instanceType
    }
}

```

- Pour plus de détails sur l'API, consultez [DescribeInstanceTypes](#) la section AWS SDK pour la référence de l'API Kotlin.

Python

SDK pour Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

class InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance
actions."""

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

```

```
        :param instance: A Boto3 Instance object. This is a high-level object
that
        wraps instance actions.
        """
        self.ec2_resource = ec2_resource
        self.instance = instance

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

    def get_instance_types(self, architecture):
        """
        Gets instance types that support the specified architecture and are
designated
        as either 'micro' or 'small'. When an instance is created, the instance
type
        you specify must support the architecture of the AMI you use.

        :param architecture: The kind of architecture the instance types must
support,
                           such as 'x86_64'.
        :return: A list of instance types that support the specified architecture
and are either 'micro' or 'small'.
        """
        try:
            inst_types = []
            it_paginator = self.ec2_resource.meta.client.get_paginator(
                "describe_instance_types"
            )
            for page in it_paginator.paginate(
                Filters=[
                    {
                        "Name": "processor-info.supported-architecture",
                        "Values": [architecture],
                    },
                    {"Name": "instance-type", "Values": ["*.micro", "*.small"]},
                ]
            ):
                inst_types += page["InstanceTypes"]
        except ClientError as err:
            logger.error(
```

```
        "Couldn't get instance types. Here's why: %s: %s",
        err.response["Error"]["Code"],
        err.response["Error"]["Message"],
    )
    raise
else:
    return inst_types
```

- Pour plus de détails sur l'API, consultez [DescribeInstanceTypes](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeInstances** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeInstances`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans les exemples de code suivants :

- [Créer et gérer un service résilient](#)
- [Démarrer avec les instances](#)

.NET

AWS SDK for .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
```

```
/// Get information about existing EC2 images.
/// </summary>
/// <returns>Async task.</returns>
public async Task DescribeInstances()
{
    // List all EC2 instances.
    await GetInstanceDescriptions();

    string tagName = "IncludeInList";
    string tagValue = "Yes";
    await GetInstanceDescriptionsFiltered(tagName, tagValue);
}

/// <summary>
/// Get information for all existing Amazon EC2 instances.
/// </summary>
/// <returns>Async task.</returns>
public async Task GetInstanceDescriptions()
{
    Console.WriteLine("Showing all instances:");
    var paginator = _amazonEC2.Paginators.DescribeInstances(new
DescribeInstancesRequest());

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

/// <summary>
/// Get information about EC2 instances filtered by a tag name and value.
/// </summary>
/// <param name="tagName">The name of the tag to filter on.</param>
/// <param name="tagValue">The value of the tag to look for.</param>
/// <returns>Async task.</returns>
public async Task GetInstanceDescriptionsFiltered(string tagName, string
tagValue)
```

```
{
    // This tag filters the results of the instance list.
    var filters = new List<Filter>
    {
        new Filter
        {
            Name = $"tag:{tagName}",
            Values = new List<string>
            {
                tagValue,
            },
        },
    };
    var request = new DescribeInstancesRequest
    {
        Filters = filters,
    };

    Console.WriteLine("\nShowing instances with tag: \"IncludeInList\" set to
    \"Yes\".");
    var paginator = _amazonEC2.Paginators.DescribeInstances(request);

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

- Pour plus de détails sur l'API, reportez-vous [DescribeInstances](#) à la section Référence des AWS SDK for .NET API.

Bash

AWS CLI avec le script Bash

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
#####
# 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

```

```
}

```

Fonctions utilitaires utilisées dans cet exemple.

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

- Pour plus de détails sur l'API, reportez-vous [DescribeInstances](#) à la section Référence des AWS CLI commandes.

C++

SDK pour C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

    for (const auto &reservation: reservations) {
```

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

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

            Aws::String typeString =

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

            Aws::String monitorString =

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

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

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

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

- Pour plus de détails sur l'API, reportez-vous [DescribeInstances](#) à la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Exemple 1 : description d'une instance

L'exemple `describe-instances` suivant décrit l'instance spécifiée.

```
aws ec2 describe-instances \
  --instance-ids i-1234567890abcdef0
```

Sortie :

```
{
  "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"
}
]
}

```

Exemple 2 : filtrage des instances correspondant au type spécifié

L'exemple `describe-instances` suivant utilise des filtres pour limiter les résultats aux instances du type spécifié.

```
aws ec2 describe-instances \
  --filters Name=instance-type,Values=m5.large
```

Pour un exemple de sortie, reportez-vous à l'exemple 1.

Pour plus d'informations, veuillez consulter la rubrique [Lister et filtrer à l'aide de la CLI](#) dans le guide de l'utilisateur Amazon EC2.

Exemple 3 : filtrage des instances correspondant au type et à la zone de disponibilité spécifiés

L'exemple `describe-instances` suivant utilise plusieurs filtres pour limiter les résultats aux instances du type spécifié qui se trouvent également dans la zone de disponibilité spécifiée.

```
aws ec2 describe-instances \
  --filters Name=instance-type,Values=t2.micro,t3.micro Name=availability-
  zone,Values=us-east-2c
```

Pour un exemple de sortie, reportez-vous à l'exemple 1.

Exemple 4 : filtrage des instances dont le type et la zone de disponibilité sont spécifiés à l'aide d'un fichier JSON

L'exemple `describe-instances` suivant utilise un fichier d'entrée JSON pour effectuer le même filtrage que dans l'exemple précédent. Lorsque les filtres deviennent plus complexes, il peut être plus facile de les spécifier dans un fichier JSON.

```
aws ec2 describe-instances \  
  --filters file://filters.json
```

Contenu de `filters.json` :

```
[  
  {  
    "Name": "instance-type",  
    "Values": ["t2.micro", "t3.micro"]  
  },  
  {  
    "Name": "availability-zone",  
    "Values": ["us-east-2c"]  
  }  
]
```

Pour un exemple de sortie, reportez-vous à l'exemple 1.

Exemple 5 : filtrage des instances comportant la balise Owner spécifiée

L'exemple `describe-instances` suivant utilise des filtres de balises pour limiter les résultats aux instances comportant une balise avec la clé de balise spécifiée (Owner), quelle que soit la valeur de la balise.

```
aws ec2 describe-instances \  
  --filters "Name=tag-key,Values=Owner"
```

Pour un exemple de sortie, reportez-vous à l'exemple 1.

Exemple 6 : filtrage des instances comportant la valeur de balise my-team spécifiée

L'exemple `describe-instances` suivant utilise des filtres de balises pour limiter les résultats aux instances comportant une balise avec la valeur de balise spécifiée (my-team), quelle que soit la clé de balise.

```
aws ec2 describe-instances \  
  --filters "Name=tag-value,Values=my-team"
```

Pour un exemple de sortie, reportez-vous à l'exemple 1.

Exemple 7 : filtrage des instances comportant la balise Owner et la valeur de balise my-team spécifiées

L'exemple `describe-instances` suivant utilise des filtres de balises pour limiter les résultats aux instances comportant la balise spécifiée (Owner=my-team).

```
aws ec2 describe-instances \  
  --filters "Name=tag:Owner,Values=my-team"
```

Pour un exemple de sortie, reportez-vous à l'exemple 1.

Exemple 8 : affichage des ID d'instance et de sous-réseau pour toutes les instances uniquement

Les exemples `describe-instances` suivants utilisent le paramètre `--query` pour n'afficher que les ID d'instance et de sous-réseau pour toutes les instances, au format JSON.

Linux et 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
```

Sortie :

```
[  
  {  
    "Instance": "i-057750d42936e468a",  
    "Subnet": "subnet-069beee9b12030077"  
  },  
  {
```

```

    "Instance": "i-001efd250faaa6ffa",
    "Subnet": "subnet-0b715c6b7db68927a"
  },
  {
    "Instance": "i-027552a73f021f3bd",
    "Subnet": "subnet-0250c25a1f4e15235"
  }
  ...
]

```

Exemple 9 : filtrage des instances correspondant au type spécifié et affichage de leurs ID d'instance uniquement

L'exemple `describe-instances` suivant utilise des filtres pour limiter les résultats aux instances du type spécifié et le paramètre `--query` pour n' afficher que les ID d'instance.

```

aws ec2 describe-instances \
  --filters "Name=instance-type,Values=t2.micro" \
  --query "Reservations[*].Instances[*].[InstanceId]" \
  --output text

```

Sortie :

```

i-031c0dc19de2fb70c
i-00d8bff789a736b75
i-0b715c6b7db68927a
i-0626d4edd54f1286d
i-00b8ae04f9f99908e
i-0fc71c25d2374130c

```

Exemple 10 : filtrage des instances correspondant au type d'instance spécifié et affichage de leur ID d'instance, de leur zone de disponibilité et de la valeur de la balise spécifiée

Les exemples `describe-instances` suivants affichent l'ID de l'instance, la zone de disponibilité et la valeur de la balise `Name` pour les instances comportant une balise nommée `tag-key`, sous forme de tableau.

Linux et 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
```

Sortie :

```
-----
|                               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   |
+-----+-----+-----+
```

Exemple 11 : description des instances d'un groupe de placement de partitions

L'exemple `describe-instances` suivant décrit l'instance spécifiée. La réponse inclut les informations sur le placement, notamment le nom du groupe de placement et le nombre de partitions correspondant à l'instance.

```
aws ec2 describe-instances \
--instance-ids i-0123a456700123456 \
--query "Reservations[*].Instances[*].Placement"
```

Sortie :

```
[
  [
    {
      "AvailabilityZone": "us-east-1c",
```

```
        "GroupName": "HDFS-Group-A",
        "PartitionNumber": 3,
        "Tenancy": "default"
    }
]
]
```

Pour plus d'informations, veuillez consulter la rubrique [Décrire des instances dans un groupe de placement](#) dans le Guide de l'utilisateur Amazon EC2.

Exemple 12 : filtrage des instances correspondant au groupe de placement et au numéro de partition spécifiés

L'exemple `describe-instances` suivant filtre les résultats uniquement sur les instances ayant le groupe de placement et le numéro de partition spécifiés.

```
aws ec2 describe-instances \
  --filters "Name=placement-group-name,Values=HDFS-Group-A" "Name=placement-
  partition-number,Values=7"
```

Seules les informations pertinentes de la sortie sont présentées ci-dessous.

```
"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"
    }
  }
]
```

```
    },  
  ],  
}
```

Pour plus d'informations, veuillez consulter la rubrique [Décrire des instances dans un groupe de placement](#) dans le Guide de l'utilisateur Amazon EC2.

Exemple 13 : filtrage des instances configurées pour autoriser l'accès aux balises à partir des métadonnées de l'instance

L'exemple `describe-instances` suivant filtre les résultats uniquement sur les instances configurées pour autoriser l'accès aux balises d'instance à partir des métadonnées de l'instance.

```
aws ec2 describe-instances \  
  --filters "Name=metadata-options.instance-metadata-tags,Values=enabled" \  
  --query "Reservations[*].Instances[*].InstanceId" \  
  --output text
```

Le résultat escompté est le suivant.

```
i-1234567890abcdefg  
i-abcdefg1234567890  
i-11111111aaaaaaaaa  
i-aaaaaaaa111111111
```

Pour plus d'informations, veuillez consulter la rubrique [Utiliser les identifications d'instance dans les métadonnées d'instance](#) dans le Guide de l'utilisateur Amazon EC2 pour les instances Linux.

- Pour plus de détails sur l'API, reportez-vous [DescribeInstances](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.paginators.DescribeInstancesIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeInstances {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        Ec2Client ec2 = Ec2Client.builder()
            .region(region)
            .build();

        describeEC2Instances(ec2);
        ec2.close();
    }

    public static void describeEC2Instances(Ec2Client ec2) {
        try {
            DescribeInstancesRequest request = DescribeInstancesRequest.builder()
                .maxResults(10)
                .build();

            DescribeInstancesIterable instancesIterable =
ec2.describeInstancesPaginator(request);
            instancesIterable.stream()
                .flatMap(r -> r.reservations().stream())
                .flatMap(reservation -> reservation.instances().stream())
                .forEach(instance -> {
                    System.out.println("Instance Id is " +
instance.instanceId());
                    System.out.println("Image id is " + instance.imageId());
                    System.out.println("Instance type is " +
instance.instanceType());
                });
        }
    }
}
```

```
        System.out.println("Instance state name is " +
instance.state().name());
        System.out.println("Monitoring information is " +
instance.monitoring().state());
    });

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

- Pour plus de détails sur l'API, reportez-vous [DescribeInstances](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

// List all of your EC2 instances running with x86_64 architecture that were
// launched this month.
export const main = async () => {
    const d = new Date();
    const year = d.getFullYear();
    const month = `0${d.getMonth() + 1}`.slice(-2);
    const launchTimePattern = `${year}-${month}-*`;
    const command = new DescribeInstancesCommand({
        Filters: [
            { Name: "architecture", Values: ["x86_64"] },
```

```
    { Name: "instance-state-name", Values: ["running"] },
    {
      Name: "launch-time",
      Values: [launchTimePattern],
    },
  ],
});

try {
  const { Reservations } = await client.send(command);
  const instanceList = Reservations.reduce((prev, current) => {
    return prev.concat(current.Instances);
  }, []);

  console.log(instanceList);
} catch (err) {
  console.error(err);
}
};
```

- Pour plus de détails sur l'API, reportez-vous [DescribeInstances](#) à la section Référence des AWS SDK for JavaScript API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun describeEC2Instances() {
  val request =
    DescribeInstancesRequest {
      maxResults = 6
    }

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

```
val response = ec2.describeInstances(request)
response.reservations?.forEach { reservation ->
    reservation.instances?.forEach { instance ->
        println("Instance Id is ${instance.instanceId}")
        println("Image id is ${instance.imageId}")
        println("Instance type is ${instance.instanceType}")
        println("Instance state name is ${instance.state?.name}")
        println("monitoring information is
${instance.monitoring?.state}")
    }
}
}
```

- Pour plus de détails sur l'API, consultez [DescribeInstances](#) la section AWS SDK pour la référence de l'API Kotlin.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit l'instance spécifiée.

```
(Get-EC2Instance -InstanceId i-12345678).Instances
```

Sortie :

```
AmiLaunchIndex      : 0
Architecture        : x86_64
BlockDeviceMappings : {/dev/sda1}
ClientToken         : T1eEy1448154045270
EbsOptimized        : False
Hypervisor          : xen
IamInstanceProfile  : Amazon.EC2.Model.IamInstanceProfile
ImageId             : ami-12345678
InstanceId          : i-12345678
InstanceLifecycle   :
InstanceType        : t2.micro
KernelId            :
KeyName             : my-key-pair
LaunchTime          : 12/4/2015 4:44:40 PM
```

```

Monitoring      : Amazon.EC2.Model.Monitoring
NetworkInterfaces : {ip-10-0-2-172.us-west-2.compute.internal}
Placement       : Amazon.EC2.Model.Placement
Platform        : Windows
PrivateDnsName  : ip-10-0-2-172.us-west-2.compute.internal
PrivateIpAddress : 10.0.2.172
ProductCodes    : {}
PublicDnsName   :
PublicIpAddress :
RamdiskId       :
RootDeviceName  : /dev/sda1
RootDeviceType  : ebs
SecurityGroups  : {default}
SourceDestCheck : True
SpotInstanceRequestId :
SriovNetSupport :
State           : Amazon.EC2.Model.InstanceState
StateReason     :
StateTransitionReason :
SubnetId        : subnet-12345678
Tags            : {Name}
VirtualizationType : hvm
VpcId           : vpc-12345678

```

Exemple 2 : Cet exemple décrit toutes vos instances dans la région actuelle, regroupées par réservation. Pour voir les détails de l'instance, étendez la collection Instances dans chaque objet de réservation.

```
Get-EC2Instance
```

Sortie :

```

GroupNames      : {}
Groups          : {}
Instances       : {}
OwnerId         : 123456789012
RequesterId     : 226008221399
ReservationId   : r-c5df370c

GroupNames      : {}
Groups          : {}
Instances       : {}
OwnerId         : 123456789012

```

```
RequesterId    : 854251627541
ReservationId  : r-63e65bab
...
```

Exemple 3 : Cet exemple illustre l'utilisation d'un filtre pour rechercher des instances EC2 dans un sous-réseau spécifique d'un VPC.

```
(Get-EC2Instance -Filter @{Name="vpc-id";Values="vpc-1a2bc34d"},@{Name="subnet-id";Values="subnet-1a2b3c4d"}).Instances
```

Sortie :

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		

- Pour plus de détails sur l'API, reportez-vous [DescribeInstances](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Python

SDK pour Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance
    actions."""

    def __init__(self, ec2_resource, instance=None):
        """
```

```
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level
resource
        is used to create additional high-level objects
        that wrap low-level Amazon EC2 service actions.
        :param instance: A Boto3 Instance object. This is a high-level object
that
        wraps instance actions.
    """
    self.ec2_resource = ec2_resource
    self.instance = instance

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

    def display(self, indent=1):
        """
        Displays information about an instance.

        :param indent: The visual indent to apply to the output.
        """
        if self.instance is None:
            logger.info("No instance to display.")
            return

        try:
            self.instance.load()
            ind = "\t" * indent
            print(f"{ind}ID: {self.instance.id}")
            print(f"{ind}Image ID: {self.instance.image_id}")
            print(f"{ind}Instance type: {self.instance.instance_type}")
            print(f"{ind}Key name: {self.instance.key_name}")
            print(f"{ind}VPC ID: {self.instance.vpc_id}")
            print(f"{ind}Public IP: {self.instance.public_ip_address}")
            print(f"{ind}State: {self.instance.state['Name']}")
        except ClientError as err:
            logger.error(
                "Couldn't display your instance. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Pour plus de détails sur l'API, consultez [DescribeInstances](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
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__
```

- Pour plus de détails sur l'API, reportez-vous [DescribeInstances](#) à la section Référence des AWS SDK for Ruby API.

Rust

SDK pour Rust

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
async fn show_state(client: &Client, ids: Option<Vec<String>>) -> Result<(),
Error> {
  let resp = client
    .describe_instances()
    .set_instance_ids(ids)
    .send()
    .await?;

  for reservation in resp.reservations() {
```

```

    for instance in reservation.instances() {
        println!("Instance ID: {}", instance.instance_id().unwrap());
        println!(
            "State:      {:?}",
            instance.state().unwrap().name().unwrap()
        );
        println!();
    }
}

Ok(())
}

```

- Pour plus de détails sur l'API, voir [DescribeInstances](#) la section de référence de l'API AWS SDK for Rust.

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

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( ).
    ENDLLOOP.
ENDLLOOP.
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.

```

- Pour plus de détails sur l'API, reportez-vous [DescribeInstances](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeInternetGateways** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeInternetGateways`.

CLI

AWS CLI

Pour décrire une passerelle Internet

L'`describe-internet-gateway` exemple suivant décrit la passerelle Internet spécifiée.

```

aws ec2 describe-internet-gateways \
  --internet-gateway-ids igw-0d0fb496b3EXAMPLE

```

Sortie :

```

{
  "InternetGateways": [
    {
      "Attachments": [
        {
          "State": "available",
          "VpcId": "vpc-0a60eb65b4EXAMPLE"
        }
      ]
    }
  ]
}

```

```

    }
  ],
  "InternetGatewayId": "igw-0d0fb496b3EXAMPLE",
  "OwnerId": "123456789012",
  "Tags": [
    {
      "Key": "Name",
      "Value": "my-igw"
    }
  ]
}
]
}

```

Pour plus d'informations, consultez [Passerelles Internet](#) dans le Guide de l'utilisateur Amazon VPC.

- Pour plus de détails sur l'API, reportez-vous [DescribeInternetGateways](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit la passerelle Internet spécifiée.

```
Get-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

Sortie :

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

Exemple 2 : Cet exemple décrit toutes vos passerelles Internet.

```
Get-EC2InternetGateway
```

Sortie :

Attachments	InternetGatewayId	Tags
-------------	-------------------	------

```
-----
{vpc-1a2b3c4d}   igw-1a2b3c4d   {}
{}               igw-2a3b4c5d   {}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeInternetGateways](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeKeyPairs** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeKeyPairs`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Démarrer avec les instances](#)

.NET

AWS SDK for .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

```

    {
        request = new DescribeKeyPairsRequest
        {
            KeyNames = new List<string> { keyPairName }
        };
    }
    var response = await _amazonEC2.DescribeKeyPairsAsync(request);
    return response.KeyPairs.ToList();
}

```

- Pour plus de détails sur l'API, reportez-vous [DescribeKeyPairs](#) à la section Référence des AWS SDK for .NET API.

Bash

AWS CLI avec le script Bash

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

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

local response

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

echo "$response"

return 0
}
```

Fonctions utilitaires utilisées dans cet exemple.

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

- Pour plus de détails sur l'API, reportez-vous [DescribeKeyPairs](#) à la section Référence des AWS CLI commandes.

C++

SDK pour C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::DescribeKeyPairsRequest request;

auto outcome = ec2Client.DescribeKeyPairs(request);
if (outcome.IsSuccess()) {
    std::cout << std::left <<
                std::setw(32) << "Name" <<
                std::setw(64) << "Fingerprint" << std::endl;

    const std::vector<Aws::EC2::Model::KeyPairInfo> &key_pairs =
        outcome.GetResult().GetKeyPairs();
    for (const auto &key_pair: key_pairs) {
        std::cout << std::left <<
                    std::setw(32) << key_pair.GetKeyName() <<
                    std::setw(64) << key_pair.GetKeyFingerprint() << std::endl;
    }
}
else {
    std::cerr << "Failed to describe key pairs:" <<
              outcome.GetError().GetMessage() << std::endl;
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeKeyPairs](#) à la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Affichage d'une paire de clés

L'exemple `describe-key-pairs` suivant affiche des informations sur la paire de clés spécifiée.

```
aws ec2 describe-key-pairs \  
  --key-names my-key-pair
```

Sortie :

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

Pour plus d'informations, veuillez consulter la rubrique [Décrire les clés publiques](#) dans le Guide de l'utilisateur Amazon EC2.

- Pour plus de détails sur l'API, reportez-vous [DescribeKeyPairs](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void describeKeys(Ec2Client ec2) {
    try {
        DescribeKeyPairsResponse response = ec2.describeKeyPairs();
        response.keyPairs().forEach(keyPair -> System.out.printf(
            "Found key pair with name %s " +
            "and fingerprint %s",
            keyPair.keyName(),
            keyPair.keyFingerprint()));
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeKeyPairs](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

export const main = async () => {
  const command = new DescribeKeyPairsCommand({});

  try {
    const { KeyPairs } = await client.send(command);
    const keyPairList = KeyPairs.map(
      (kp) => ` • ${kp.KeyPairId}: ${kp.KeyName}`,
    ).join("\n");
    console.log("The following key pairs were found in your account:");
    console.log(keyPairList);
  } catch (err) {
    console.error(err);
  }
};
```

- Pour plus de détails sur l'API, reportez-vous [DescribeKeyPairs](#) à la section Référence des AWS SDK for JavaScript API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun describeEC2Keys() {
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeKeyPairs(DescribeKeyPairsRequest {})
        response.keyPairs?.forEach { keyPair ->
            println("Found key pair with name ${keyPair.keyName} and fingerprint
            ${ keyPair.keyFingerprint}")
        }
    }
}
```

- Pour plus de détails sur l'API, consultez [DescribeKeyPairs](#) la section AWS SDK pour la référence de l'API Kotlin.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit la paire de clés spécifiée.

```
Get-EC2KeyPair -KeyName my-key-pair
```

Sortie :

```
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
```

Exemple 2 : Cet exemple décrit toutes vos paires de clés.

```
Get-EC2KeyPair
```

- Pour plus de détails sur l'API, reportez-vous [DescribeKeyPairs](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Python

SDK pour Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

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

    def list(self, limit):
        """
        Displays a list of key pairs for the current account.

        :param limit: The maximum number of key pairs to list.
        """
        try:
            for kp in self.ec2_resource.key_pairs.limit(limit):
                print(f"Found {kp.key_type} key {kp.name} with fingerprint:")
                print(f"\t{kp.key_fingerprint}")
        except ClientError as err:
            logger.error(
                "Couldn't list key pairs. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Pour plus de détails sur l'API, consultez [DescribeKeyPairs](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
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.
```

- Pour plus de détails sur l'API, reportez-vous [DescribeKeyPairs](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeNetworkAcls** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeNetworkAcls`.

CLI

AWS CLI

Pour décrire les ACL de votre réseau

L'`describe-network-acls` suivant permet de récupérer des informations sur les ACL de votre réseau.

```
aws ec2 describe-network-acls
```

Sortie :

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

Pour plus d'informations, consultez la section [ACL réseau](#) dans le guide de l'AWS utilisateur du VPC.

- Pour plus de détails sur l'API, reportez-vous [DescribeNetworkAcls](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit l'ACL réseau spécifiée.

```
Get-EC2NetworkAcl -NetworkAclId acl-12345678
```

Sortie :

```
Associations : {aclassoc-1a2b3c4d}
Entries      : {Amazon.EC2.Model.NetworkAclEntry,
               Amazon.EC2.Model.NetworkAclEntry}
IsDefault    : False
NetworkAclId : acl-12345678
Tags         : {Name}
VpcId        : vpc-12345678
```

Exemple 2 : Cet exemple décrit les règles pour l'ACL réseau spécifiée.

```
(Get-EC2NetworkAcl -NetworkAclId acl-12345678).Entries
```

Sortie :

```
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
```

Exemple 3 : Cet exemple décrit toutes les ACL de votre réseau.

```
Get-EC2NetworkACL
```

- Pour plus de détails sur l'API, reportez-vous [DescribeNetworkAcls](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeNetworkInterfaceAttribute** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeNetworkInterfaceAttribute`.

CLI

AWS CLI

Pour décrire l'attribut de pièce jointe d'une interface réseau

Cet exemple de commande décrit l'attachmentattribut de l'interface réseau spécifiée.

Commande :

```
aws ec2 describe-network-interface-attribute --network-interface-id eni-686ea200 --attribute attachment
```

Sortie :

```
{
  "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"  
  }  
}
```

Pour décrire l'attribut de description d'une interface réseau

Cet exemple de commande décrit l'`description` attribut de l'interface réseau spécifiée.

Commande :

```
aws ec2 describe-network-interface-attribute --network-interface-id eni-686ea200  
--attribute description
```

Sortie :

```
{  
  "NetworkInterfaceId": "eni-686ea200",  
  "Description": {  
    "Value": "My description"  
  }  
}
```

Pour décrire l'attribut GroupSet d'une interface réseau

Cet exemple de commande décrit l'`groupSet` attribut de l'interface réseau spécifiée.

Commande :

```
aws ec2 describe-network-interface-attribute --network-interface-id eni-686ea200  
--attribute groupSet
```

Sortie :

```
{  
  "NetworkInterfaceId": "eni-686ea200",  
  "Groups": [  
    {  
      "GroupName": "my-security-group",  
      "GroupId": "sg-903004f8"  
    }  
  ]  
}
```

```
}
```

Pour décrire l' `sourceDestCheck` attribut d'une interface réseau

Cet exemple de commande décrit l'`sourceDestCheck` attribut de l'interface réseau spécifiée.

Commande :

```
aws ec2 describe-network-interface-attribute --network-interface-id eni-686ea200
--attribute sourceDestCheck
```

Sortie :

```
{
  "NetworkInterfaceId": "eni-686ea200",
  "SourceDestCheck": {
    "Value": true
  }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeNetworkInterfaceAttribute](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit l'interface réseau spécifiée.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
Attachment
```

Sortie :

```
Attachment           : Amazon.EC2.Model.NetworkInterfaceAttachment
```

Exemple 2 : Cet exemple décrit l'interface réseau spécifiée.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
Description
```

Sortie :

```
Description      : My description
```

Exemple 3 : Cet exemple décrit l'interface réseau spécifiée.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
GroupSet
```

Sortie :

```
Groups           : {my-security-group}
```

Exemple 4 : Cet exemple décrit l'interface réseau spécifiée.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
SourceDestCheck
```

Sortie :

```
SourceDestCheck  : True
```

- Pour plus de détails sur l'API, reportez-vous [DescribeNetworkInterfaceAttribute](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeNetworkInterfaces** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeNetworkInterfaces`.

CLI

AWS CLI

Pour décrire vos interfaces réseau

Cet exemple décrit toutes vos interfaces réseau.

Commande :

```
aws ec2 describe-network-interfaces
```

Sortie :

```
{
  "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"
}
]
```

Cet exemple décrit les interfaces réseau dotées d'une balise avec la clé `Purpose` et la valeur `Prod`.

Commande :

```
aws ec2 describe-network-interfaces --filters Name=tag:Purpose,Values=Prod
```

Sortie :

```
{
  "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"
}
]
```

- Pour plus de détails sur l'API, reportez-vous [DescribeNetworkInterfaces](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit l'interface réseau spécifiée.

```
Get-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

Sortie :

```
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
```

Exemple 2 : Cet exemple décrit toutes vos interfaces réseau.

```
Get-EC2NetworkInterface
```

- Pour plus de détails sur l'API, reportez-vous [DescribeNetworkInterfaces](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribePlacementGroups** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribePlacementGroups`.

CLI

AWS CLI

Pour décrire vos groupes de placement

Cet exemple de commande décrit tous vos groupes de placement.

Commande :

```
aws ec2 describe-placement-groups
```

Sortie :

```
{
  "PlacementGroups": [
    {
      "GroupName": "my-cluster",
      "State": "available",
      "Strategy": "cluster"
    },
    ...
  ]
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribePlacementGroups](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit le groupe de placement spécifié.

```
Get-EC2PlacementGroup -GroupName my-placement-group
```

Sortie :

GroupName	State	Strategy
-----	-----	-----
my-placement-group	available	cluster

- Pour plus de détails sur l'API, reportez-vous [DescribePlacementGroups](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation `DescribePrefixLists` avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribePrefixLists`.

CLI

AWS CLI

Pour décrire les listes de préfixes

Cet exemple répertorie toutes les listes de préfixes disponibles pour la région.

Commande :

```
aws ec2 describe-prefix-lists
```

Sortie :

```
{
  "PrefixLists": [
    {
      "PrefixListName": "com.amazonaws.us-east-1.s3",
      "Cidrs": [
        "54.231.0.0/17"
      ],
      "PrefixListId": "pl-63a5400a"
    }
  ]
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribePrefixLists](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple extrait le fichier disponible AWS services sous forme de liste de préfixes pour la région

```
Get-EC2PrefixList
```

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [DescribePrefixLists](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation `DescribeRegions` avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeRegions`.

C++**Kit de développement logiciel (SDK) for C++****Note**

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

Aws::EC2::Model::DescribeRegionsRequest request;
auto outcome = ec2Client.DescribeRegions(request);
bool result = true;
if (outcome.IsSuccess()) {
    std::cout << std::left <<
                std::setw(32) << "RegionName" <<
                std::setw(64) << "Endpoint" << std::endl;

    const auto &regions = outcome.GetResult().GetRegions();
}
```

```
for (const auto &region: regions) {
    std::cout << std::left <<
        std::setw(32) << region.GetRegionName() <<
        std::setw(64) << region.GetEndpoint() << std::endl;
}
else {
    std::cerr << "Failed to describe regions:" <<
        outcome.GetError().GetMessage() << std::endl;
    result = false;
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeRegions](#) à la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Exemple 1 : description de l'ensemble de vos régions activées

L'exemple `describe-regions` suivant décrit toutes les régions activées pour votre compte.

```
aws ec2 describe-regions
```

Sortie :

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

Pour plus d'informations, veuillez consulter la rubrique [Régions et zones](#) dans le Guide de l'utilisateur Amazon EC2.

Exemple 2 : description des régions activées avec un point de terminaison dont le nom contient une chaîne de caractères spécifique

L'exemple `describe-regions` suivant décrit toutes les régions que vous avez activées et dont le point de terminaison contient la chaîne « us ».

```
aws ec2 describe-regions \
  --filters "Name=endpoint,Values=*us*"
```

Sortie :

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

Pour plus d'informations, veuillez consulter la rubrique [Régions et zones](#) dans le Guide de l'utilisateur Amazon EC2.

Exemple 3 : description de toutes les régions

L'exemple `describe-regions` suivant décrit toutes les régions disponibles, y compris les régions désactivées.

```
aws ec2 describe-regions \
  --all-regions
```

Sortie :

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

Pour plus d'informations, veuillez consulter la rubrique [Régions et zones](#) dans le Guide de l'utilisateur Amazon EC2.

Exemple 4 : énumération des noms de région uniquement

L'exemple `describe-regions` suivant utilise le paramètre `--query` pour filtrer la sortie et ne renvoyer que les noms des régions sous forme de texte.

```
aws ec2 describe-regions \  
  --all-regions \  
  --query "Regions[].{Name:RegionName}" \  
  --output text
```

Sortie :

```
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
```

Pour plus d'informations, veuillez consulter la rubrique [Régions et zones](#) dans le Guide de l'utilisateur Amazon EC2.

- Pour plus de détails sur l'API, reportez-vous [DescribeRegions](#) à la section Référence des AWS CLI commandes.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

export const main = async () => {
  const command = new DescribeRegionsCommand({
    // By default this command will not show regions that require you to opt-in.
    // When AllRegions true even the regions that require opt-in will be
    returned.
    AllRegions: true,
    // You can omit the Filters property if you want to get all regions.
    Filters: [
      {
        Name: "region-name",
        // You can specify multiple values for a filter.
        // You can also use '*' as a wildcard. This will return all
        // of the regions that start with `us-east-`.
        Values: ["ap-southeast-4"],
      },
    ],
  });

  try {
    const { Regions } = await client.send(command);
    const regionsList = Regions.map((reg) => ` • ${reg.RegionName}`);
    console.log("Found regions:");
    console.log(regionsList.join("\n"));
  } catch (err) {
```

```
    console.error(err);
  }
};
```

- Pour plus de détails sur l'API, reportez-vous [DescribeRegions](#) à la section Référence des AWS SDK for JavaScript API.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit les régions qui sont disponibles pour vous.

```
Get-EC2Region
```

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [DescribeRegions](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Ruby

Kit SDK pour Ruby

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
require "aws-sdk-ec2"

# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   list_regions_endpoints(Aws::EC2::Client.new(region: 'us-west-2'))
def list_regions_endpoints(ec2_client)
  result = ec2_client.describe_regions
  # Enable pretty printing.
  max_region_string_length = 16
  max_endpoint_string_length = 33
  # Print header.
  print "Region"
  print " " * (max_region_string_length - "Region".length)
  print "  Endpoint\n"
  print "-" * max_region_string_length
  print " "
  print "-" * max_endpoint_string_length
  print "\n"
  # Print Regions and their endpoints.
  result.regions.each do |region|
    print region.region_name
    print " " * (max_region_string_length - region.region_name.length)
    print " "
    print region.endpoint
    print "\n"
  end
end

# Displays a list of Amazon Elastic Compute Cloud (Amazon EC2)
# Availability Zones available to you depending on the AWS Region
# of the Amazon EC2 client.
```

```
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
# list_availability_zones(Aws::EC2::Client.new(region: 'us-west-2'))
def list_availability_zones(ec2_client)
  result = ec2_client.describe_availability_zones
  # Enable pretty printing.
  max_region_string_length = 16
  max_zone_string_length = 18
  max_state_string_length = 9
  # Print header.
  print "Region"
  print " " * (max_region_string_length - "Region".length)
  print " Zone"
  print " " * (max_zone_string_length - "Zone".length)
  print " State\n"
  print "-" * max_region_string_length
  print " "
  print "-" * max_zone_string_length
  print " "
  print "-" * max_state_string_length
  print "\n"
  # Print Regions, Availability Zones, and their states.
  result.availability_zones.each do |zone|
    print zone.region_name
    print " " * (max_region_string_length - zone.region_name.length)
    print " "
    print zone.zone_name
    print " " * (max_zone_string_length - zone.zone_name.length)
    print " "
    print zone.state
    # Print any messages for this Availability Zone.
    if zone.messages.count.positive?
      print "\n"
      puts " Messages for this zone:"
      zone.messages.each do |message|
        print "   #{message.message}\n"
      end
    end
    print "\n"
  end
end
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__
```

- Pour plus de détails sur l'API, reportez-vous [DescribeRegions](#) à la section Référence des AWS SDK for Ruby API.

Rust

SDK pour Rust

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

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(())
}

```

- Pour plus de détails sur l'API, voir [DescribeRegions](#) la section de référence de l'API AWS SDK for Rust.

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

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.

```

- Pour plus de détails sur l'API, reportez-vous [DescribeRegions](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeRouteTables** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeRouteTables`.

CLI

AWS CLI

Pour décrire vos tables de routage

L'`describe-route-tables` exemple suivant permet de récupérer les détails de vos tables de routage.

```
aws ec2 describe-route-tables
```

Sortie :

```
{
  "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"
  }
]
}

```

Pour plus d'informations, consultez la section [Utilisation des tables de routage](#) dans le guide de l'utilisateur AWS VPC.

- Pour plus de détails sur l'API, reportez-vous [DescribeRouteTables](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit toutes vos tables de routage.

```
Get-EC2RouteTable
```

Sortie :

```
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 :

```

Exemple 2 : Cet exemple renvoie les détails de la table de routage spécifiée.

```
Get-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

Exemple 3 : Cet exemple décrit les tables de routage pour le VPC spécifié.

```
Get-EC2RouteTable -Filter @{ Name="vpc-id"; Values="vpc-1a2b3c4d" }
```

Sortie :

```

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

```

- Pour plus de détails sur l'API, reportez-vous [DescribeRouteTables](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation `DescribeScheduledInstanceAvailability` avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeScheduledInstanceAvailability`.

CLI

AWS CLI

Pour décrire un calendrier disponible

Cet exemple décrit un calendrier qui a lieu chaque semaine le dimanche, à compter de la date spécifiée.

Commande :

```
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
```

Sortie :

```
{
  "ScheduledInstanceAvailabilitySet": [
    {
      "AvailabilityZone": "us-west-2b",
      "TotalScheduledInstanceHours": 1219,
      "PurchaseToken": "eyJ2IjoiMSIsInMiOiJEsImMiOi...",
      "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"
    },
    ...
]
}

```

Pour affiner les résultats, vous pouvez ajouter des filtres qui spécifient le système d'exploitation, le réseau et le type d'instance.

Commande :

```
--filters Nom = plate-forme, valeurs = nom Linux/UNIX = plate-forme réseau, valeurs = nom
EC2-VPC = type d'instance, valeurs = C4.large
```

- Pour plus de détails sur l'API, reportez-vous [DescribeScheduledInstanceAvailability](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit un calendrier qui a lieu chaque semaine le dimanche, à compter de la date spécifiée.

```

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

```

Sortie :

```

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
...
```

Exemple 2 : pour affiner les résultats, vous pouvez ajouter des filtres pour des critères tels que le système d'exploitation, le réseau et le type d'instance.

```
-Filter @{ Name="platform";Values="Linux/UNIX" },@{ Name="network-
platform";Values="EC2-VPC" },@{ Name="instance-type";Values="c4.large" }
```

- Pour plus de détails sur l'API, reportez-vous [DescribeScheduledInstanceAvailability](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeScheduledInstances** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeScheduledInstances`.

CLI

AWS CLI

Pour décrire vos instances planifiées

Cet exemple décrit l'instance planifiée spécifiée.

Commande :

```
aws ec2 describe-scheduled-instances --scheduled-instance-ids
sci-1234-1234-1234-1234-123456789012
```

Sortie :

```
{
  "ScheduledInstanceSet": [
    {
      "AvailabilityZone": "us-west-2b",
      "ScheduledInstanceId": "sci-1234-1234-1234-1234-123456789012",
      "HourlyPrice": "0.095",
      "CreateDate": "2016-01-25T21:43:38.612Z",
      "Recurrence": {
        "OccurrenceDaySet": [
          1
        ],
        "Interval": 1,
        "Frequency": "Weekly",
        "OccurrenceRelativeToEnd": false,
        "OccurrenceUnit": ""
      },
      "Platform": "Linux/UNIX",
      "TermEndDate": "2017-01-31T09:00:00Z",
      "InstanceCount": 1,
      "SlotDurationInHours": 32,
      "TermStartDate": "2016-01-31T09:00:00Z",
      "NetworkPlatform": "EC2-VPC",
      "TotalScheduledInstanceHours": 1696,
      "NextSlotStartTime": "2016-01-31T09:00:00Z",
      "InstanceType": "c4.large"
    }
  ]
}
```

Cet exemple décrit toutes vos instances planifiées.

Commande :

```
aws ec2 describe-scheduled-instances
```

- Pour plus de détails sur l'API, reportez-vous [DescribeScheduledInstances](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit l'instance planifiée spécifiée.

```
Get-EC2ScheduledInstance -ScheduledInstanceId  
sci-1234-1234-1234-1234-123456789012
```

Sortie :

```
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
```

Exemple 2 : Cet exemple décrit toutes vos instances planifiées.

```
Get-EC2ScheduledInstance
```

- Pour plus de détails sur l'API, reportez-vous [DescribeScheduledInstances](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeSecurityGroups** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeSecurityGroups`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Démarrer avec les instances](#)

.NET

AWS SDK for .NET

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Retrieve information for an Amazon EC2 security group.
/// </summary>
/// <param name="groupId">The Id of the Amazon EC2 security group.</param>
/// <returns>A list of security group information.</returns>
public async Task<List<SecurityGroup>> DescribeSecurityGroups(string groupId)
{
    var request = new DescribeSecurityGroupsRequest();
    var groupIds = new List<string> { groupId };
    request.GroupIds = groupIds;

    var response = await _amazonEC2.DescribeSecurityGroupsAsync(request);
    return response.SecurityGroups;
}

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

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

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

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

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

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

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

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

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

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

- Pour plus de détails sur l'API, reportez-vous [DescribeSecurityGroups](#) à la section Référence des AWS SDK for .NET API.

Bash

AWS CLI avec le script Bash

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

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

```

```

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

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

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

local error_code=${?}

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

echo "$response"

return 0
}

```

Fonctions utilitaires utilisées dans cet exemple.

```

#####
# 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
}

```

- Pour plus de détails sur l'API, reportez-vous [DescribeSecurityGroups](#) à la section Référence des AWS CLI commandes.

C++

SDK pour C++

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::DescribeSecurityGroupsRequest request;

if (!groupID.empty()) {
    request.AddGroupIds(groupID);
}

Aws::String nextToken;
do {
    if (!nextToken.empty()) {
        request.SetNextToken(nextToken);
    }

    auto outcome = ec2Client.DescribeSecurityGroups(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left <<
            std::setw(32) << "Name" <<
            std::setw(30) << "GroupId" <<
            std::setw(30) << "VpcId" <<
            std::setw(64) << "Description" << std::endl;

        const std::vector<Aws::EC2::Model::SecurityGroup> &securityGroups =
            outcome.GetResult().GetSecurityGroups();

        for (const auto &securityGroup: securityGroups) {
            std::cout << std::left <<
                std::setw(32) << securityGroup.GetGroupName() <<
                std::setw(30) << securityGroup.GetGroupId() <<
                std::setw(30) << securityGroup.GetVpcId() <<
                std::setw(64) << securityGroup.GetDescription() <<
                std::endl;
        }
    }
}
```

```
    }
    else {
        std::cerr << "Failed to describe security groups:" <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    nextToken = outcome.GetResult().GetNextToken();
} while (!nextToken.empty());
```

- Pour plus de détails sur l'API, reportez-vous [DescribeSecurityGroups](#) à la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Exemple 1 : description d'un groupe de sécurité

L'exemple `describe-security-groups` suivant décrit le groupe de sécurité spécifié.

```
aws ec2 describe-security-groups \
  --group-ids sg-903004f8
```

Sortie :

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

Exemple 2 : description des groupes de sécurité comportant des règles spécifiques

L'exemple `describe-security-groups` suivant utilise des filtres pour étendre les résultats aux groupes de sécurité dotés d'une règle autorisant le trafic SSH (port 22) et d'une règle autorisant le trafic provenant de toutes les adresses (`0.0.0.0/0`). L'exemple utilise le paramètre `--query` pour n'afficher que les noms des groupes de sécurité. Les groupes de sécurité doivent correspondre à tous les filtres pour être retournés dans les résultats ; toutefois, une règle unique ne doit pas nécessairement correspondre à tous les filtres. Par exemple, la sortie renvoie un groupe de sécurité avec une règle qui autorise le trafic SSH depuis une adresse IP spécifique et une autre règle qui autorise le trafic HTTP depuis toutes les adresses.

```
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
```

Sortie :

```
default
my-security-group
web-servers
launch-wizard-1
```

Exemple 3 : description des groupes de sécurité sur la base de balises

L'exemple `describe-security-groups` suivant utilise des filtres pour limiter les résultats aux groupes de sécurité incluant `test` dans le nom du groupe de sécurité et comportant la balise `Test=To-delete`. L'exemple utilise le paramètre `--query` pour n'afficher que les noms et les identifiants des groupes de sécurité.

```
aws ec2 describe-security-groups \
  --filters Name=group-name,Values=*test* Name=tag:Test,Values=To-delete \
  --query "SecurityGroups[*].{Name:GroupName, ID:GroupId}"
```

Sortie :

```
[
  {
    "Name": "testfornewinstance",
```

```
    "ID": "sg-33bb22aa"
  },
  {
    "Name": "newgroupptest",
    "ID": "sg-1a2b3c4d"
  }
]
```

Pour d'autres exemples d'utilisation de filtres de balises, veuillez consulter la rubrique [Utiliser des balises](#) dans le Guide de l'utilisateur Amazon EC2.

- Pour plus de détails sur l'API, reportez-vous [DescribeSecurityGroups](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
    try {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupIds(groupId)
            .build();

        // Use a paginator.
        DescribeSecurityGroupsIterable listGroups =
ec2.describeSecurityGroupsPaginator(request);
        listGroups.stream()
            .flatMap(r -> r.securityGroups().stream())
            .forEach(group -> System.out
                .println(" Group id: " +group.groupId() + " group name = " +
group.groupName()));
    } catch (Ec2Exception e) {
```

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

- Pour plus de détails sur l'API, reportez-vous [DescribeSecurityGroups](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

// Log the details of a specific security group.
export const main = async () => {
    const command = new DescribeSecurityGroupsCommand({
        GroupIds: ["SECURITY_GROUP_ID"],
    });

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

- Pour plus de détails sur l'API, reportez-vous [DescribeSecurityGroups](#) à la section Référence des AWS SDK for JavaScript API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

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

- Pour plus de détails sur l'API, consultez [DescribeSecurityGroups](#) la section AWS SDK pour la référence de l'API Kotlin.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit le groupe de sécurité spécifié pour un VPC. Lorsque vous travaillez avec des groupes de sécurité appartenant à un VPC, vous devez utiliser l'ID du groupe de sécurité (GroupId paramètre -), et non le nom (-GroupName paramètre), pour référencer le groupe.

```
Get-EC2SecurityGroup -GroupId sg-12345678
```

Sortie :

```
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
```

Exemple 2 : Cet exemple décrit le groupe de sécurité spécifié pour EC2-Classic. Lorsque vous travaillez avec des groupes de sécurité pour EC2-Classic, vous pouvez utiliser le nom du groupe (- GroupName paramètre) ou l'ID du groupe (- GroupId paramètre) pour référencer le groupe de sécurité.

```
Get-EC2SecurityGroup -GroupName my-security-group
```

Sortie :

```
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           :
```

Exemple 3 : Cet exemple récupère tous les groupes de sécurité pour le fichier vpc-0fc1ff23456b789eb

```
Get-EC2SecurityGroup -Filter @{Name="vpc-id";Values="vpc-0fc1ff23456b789eb"}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeSecurityGroups](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Python

SDK pour Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

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

    def describe(self):
        """
        Displays information about the security group.
        """
        if self.security_group is None:
            logger.info("No security group to describe.")
            return

        try:
```

```

print(f"Security group: {self.security_group.group_name}")
print(f"\tID: {self.security_group.id}")
print(f"\tVPC: {self.security_group.vpc_id}")
if self.security_group.ip_permissions:
    print(f"Inbound permissions:")
    pp(self.security_group.ip_permissions)
except ClientError as err:
    logger.error(
        "Couldn't get data for security group %s. Here's why: %s: %s",
        self.security_group.id,
        err.response["Error"]["Code"],
        err.response["Error"]["Message"],
    )
    raise

```

- Pour plus de détails sur l'API, consultez [DescribeSecurityGroups](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

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.
```

- Pour plus de détails sur l'API, reportez-vous [DescribeSecurityGroups](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeSnapshotAttribute** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeSnapshotAttribute`.

CLI

AWS CLI

Pour décrire les attributs d'un instantané

L'`describe-snapshot-attribute` exemple suivant répertorie les comptes avec lesquels un instantané est partagé.

```
aws ec2 describe-snapshot-attribute \  
  --snapshot-id snap-01234567890abcdef \  
  --attribute createVolumePermission
```

Sortie :

```
{
  "SnapshotId": "snap-01234567890abcdef",
  "CreateVolumePermissions": [
    {
      "UserId": "123456789012"
    }
  ]
}
```

Pour plus d'informations, consultez [Partager un instantané Amazon EBS](#) dans le guide de l'utilisateur d'Amazon Elastic Compute Cloud.

- Pour plus de détails sur l'API, reportez-vous [DescribeSnapshotAttribute](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit l'attribut spécifié du cliché spécifié.

```
Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute ProductCodes
```

Sortie :

CreateVolumePermissions	ProductCodes	SnapshotId
{}	{}	snap-12345678

Exemple 2 : Cet exemple décrit l'attribut spécifié du cliché spécifié.

```
(Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute CreateVolumePermission).CreateVolumePermissions
```

Sortie :

Group	UserId
all	

- Pour plus de détails sur l'API, reportez-vous [DescribeSnapshotAttribute](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeSnapshots** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeSnapshots`.

CLI

AWS CLI

Exemple 1 : description d'un instantané

L'exemple `describe-snapshots` suivant décrit l'instantané spécifié.

```
aws ec2 describe-snapshots \
  --snapshot-ids snap-1234567890abcdef0
```

Sortie :

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

Pour plus d'informations, veuillez consulter la rubrique [Instantanés Amazon EBS](#) dans le Guide de l'utilisateur Amazon EC2.

Exemple 2 : description d'instantanés sur la base de filtres

L'`describe-snapshots` suivant utilise des filtres pour étendre les résultats aux instantanés appartenant à votre AWS compte qui se trouvent dans l'`pending` État. L'exemple utilise le paramètre `--query` pour n'afficher que les identifiants des instantanés et l'heure à laquelle ils ont été démarrés.

```
aws ec2 describe-snapshots \  
  --owner-ids self \  
  --filters Name=status,Values=pending \  
  --query "Snapshots[*].{ID:SnapshotId,Time:StartTime}"
```

Sortie :

```
[  
  {  
    "ID": "snap-1234567890abcdef0",  
    "Time": "2019-08-04T12:48:18.000Z"  
  },  
  {  
    "ID": "snap-066877671789bd71b",  
    "Time": "2019-08-04T02:45:16.000Z"  
  },  
  ...  
]
```

L'exemple `describe-snapshots` suivant utilise des filtres pour limiter les résultats aux instantanés créés à partir du volume spécifié. L'exemple utilise le paramètre `--query` pour n'afficher que les ID des instantanés.

```
aws ec2 describe-snapshots \  
  --filters Name=volume-id,Values=049df61146c4d7901 \  
  --query "Snapshots[*].[SnapshotId]" \  
  --output text
```

Sortie :

```
snap-1234567890abcdef0  
snap-08637175a712c3fb9  
...
```

Pour d'autres exemples d'utilisation de filtres, veuillez consulter la rubrique [Lister et filtrer vos ressources](#) dans le Guide de l'utilisateur Amazon EC2.

Exemple 3 : description d'instantanés sur la base de balises

L'exemple `describe-snapshots` suivant utilise des filtres de balises pour limiter les résultats aux instantanés comportant la balise `Stack=Prod`.

```
aws ec2 describe-snapshots \  
  --filters Name=tag:Stack,Values=prod
```

Pour obtenir un exemple de la manière de procéder pour `describe-snapshots`, reportez-vous à l'exemple 1.

Pour d'autres exemples d'utilisation de filtres de balises, veuillez consulter la rubrique [Utiliser des balises](#) dans le Guide de l'utilisateur Amazon EC2.

Exemple 4 : description d'instantanés en fonction de l'âge

L'exemple `describe-snapshotsexemple` suivant utilise des expressions JMESpath pour décrire tous les instantanés créés par votre AWS compte avant la date spécifiée. Il n'affiche que les identifiants des instantanés.

```
aws ec2 describe-snapshots \  
  --owner-ids 012345678910 \  
  --query "Snapshots[?(StartTime<='2020-03-31')].[SnapshotId]"
```

Pour d'autres exemples d'utilisation de filtres, veuillez consulter la rubrique [Lister et filtrer vos ressources](#) dans le Guide de l'utilisateur Amazon EC2.

Exemple 5 : affichage des instantanés archivés uniquement

L'exemple `describe-snapshots` suivant ne liste que les instantanés qui sont stockés dans le niveau d'archivage spécifié.

```
aws ec2 describe-snapshots \  
  --filters "Name=storage-tier,Values=archive"
```

Sortie :

```
{
```

```
"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": []  
  },  
]  
}
```

Pour plus d'informations, veuillez consulter la rubrique [Afficher les instantanés archivés](#) dans le Guide de l'utilisateur Amazon Elastic Compute Cloud.

- Pour plus de détails sur l'API, reportez-vous [DescribeSnapshots](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit le cliché spécifié.

```
Get-EC2Snapshot -SnapshotId snap-12345678
```

Sortie :

```
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
```

Exemple 2 : Cet exemple décrit les instantanés dotés d'une balise « Nom ».

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" }
```

Exemple 3 : Cet exemple décrit les instantanés dotés d'une balise « Name » avec la valeur « TestValue ».

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" -and
  $_.Tags.Value -eq "TestValue" }
```

Exemple 4 : Cet exemple décrit tous vos instantanés.

```
Get-EC2Snapshot -Owner self
```

- Pour plus de détails sur l'API, reportez-vous [DescribeSnapshots](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Rust

SDK pour Rust

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Indique l'état d'un instantané.

```
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(())
}

```

- Pour plus de détails sur l'API, voir [DescribeSnapshots](#) la section de référence de l'API AWS SDK for Rust.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation `DescribeSpotDatafeedSubscription` avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeSpotDatafeedSubscription`.

CLI

AWS CLI

Pour décrire l'abonnement au flux de données d'une instance Spot pour un compte

Cet exemple de commande décrit le flux de données du compte.

Commande :

```
aws ec2 describe-spot-datafeed-subscription
```

Sortie :

```
{
  "SpotDatafeedSubscription": {
    "OwnerId": "123456789012",
    "Prefix": "spotdata",
    "Bucket": "my-s3-bucket",
    "State": "Active"
  }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeSpotDatafeedSubscription](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit le flux de données de votre instance Spot.

```
Get-EC2SpotDatafeedSubscription
```

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [DescribeSpotDatafeedSubscription](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeSpotFleetInstances** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeSpotFleetInstances`.

CLI

AWS CLI

Pour décrire les instances Spot associées à un parc Spot

Cet exemple de commande répertorie les instances Spot associées au parc Spot spécifié.

Commande :

```
aws ec2 describe-spot-fleet-instances --spot-fleet-request-id sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
```

Sortie :

```
{
  "ActiveInstances": [
    {
```

```

        "InstanceId": "i-1234567890abcdef0",
        "InstanceType": "m3.medium",
        "SpotInstanceRequestId": "sir-08b93456"
    },
    ...
],
"SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE"
}

```

- Pour plus de détails sur l'API, reportez-vous [DescribeSpotFleetInstances](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit les instances associées à la demande de parc Spot spécifiée.

```
Get-EC2SpotFleetInstance -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
```

Sortie :

InstanceId	InstanceType	SpotInstanceRequestId
i-f089262a	c3.large	sir-12345678
i-7e8b24a4	c3.large	sir-87654321

- Pour plus de détails sur l'API, reportez-vous [DescribeSpotFleetInstances](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeSpotFleetRequestHistory** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeSpotFleetRequestHistory`.

CLI

AWS CLI

Pour décrire l'historique de la flotte Spot

Cet exemple de commande renvoie l'historique du parc de Spot spécifié à partir de l'heure spécifiée.

Commande :

```
aws ec2 describe-spot-fleet-request-history --spot-fleet-request-id sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE --start-time 2015-05-26T00:00:00Z
```

L'exemple de sortie suivant montre les lancements réussis de deux instances Spot pour le parc Spot.

Sortie :

```
{
  "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"
}

```

- Pour plus de détails sur l'API, reportez-vous [DescribeSpotFleetRequestHistory](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit l'historique de la demande de parc Spot spécifiée.

```
Get-EC2SpotFleetRequestHistory -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE -StartTime 2015-12-26T00:00:00Z
```

Sortie :

```

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
```

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [DescribeSpotFleetRequestHistory](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeSpotFleetRequests** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeSpotFleetRequests`.

CLI

AWS CLI

Pour décrire vos demandes relatives à une flotte Spot

Cet exemple décrit toutes vos demandes de flotte Spot.

Commande :

```
aws ec2 describe-spot-fleet-requests
```

Sortie :

```
{
  "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"
}
]
}

```

Pour décrire une demande de flotte Spot

Cet exemple décrit la demande de parc Spot spécifiée.

Commande :

```
aws ec2 describe-spot-fleet-requests --spot-fleet-request-ids sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
```

Sortie :

```

{
  "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"
}
]
}

```

- Pour plus de détails sur l'API, reportez-vous [DescribeSpotFleetRequests](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit la demande de parc Spot spécifiée.

```
Get-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE | format-list
```

Sortie :

```
ConfigData      : Amazon.EC2.Model.SpotFleetRequestConfigData
CreateTime      : 12/26/2015 8:23:33 AM
SpotFleetRequestId : sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
SpotFleetRequestState : active
```

Exemple 2 : Cet exemple décrit toutes vos demandes de flotte Spot.

```
Get-EC2SpotFleetRequest
```

- Pour plus de détails sur l'API, reportez-vous [DescribeSpotFleetRequests](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeSpotInstanceRequests** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeSpotInstanceRequests`.

CLI

AWS CLI

Exemple 1 : Pour décrire une demande d'instance Spot

L'`describe-spot-instance-requestsexemple` suivant décrit la demande d'instance Spot spécifiée.

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

Sortie :

```
{
  "SpotInstanceRequests": [
    {
      "CreateTime": "2018-04-30T18:14:55.000Z",
      "InstanceId": "i-1234567890abcdef1",
```

```
"LaunchSpecification": {
  "InstanceType": "t2.micro",
  "ImageId": "ami-003634241a8fcdec0",
  "KeyName": "my-key-pair",
  "SecurityGroups": [
    {
      "GroupName": "default",
      "GroupId": "sg-e38f24a7"
    }
  ],
  "BlockDeviceMappings": [
    {
      "DeviceName": "/dev/sda1",
      "Ebs": {
        "DeleteOnTermination": true,
        "SnapshotId": "snap-0e54a519c999adbbd",
        "VolumeSize": 8,
        "VolumeType": "standard",
        "Encrypted": false
      }
    }
  ],
  "NetworkInterfaces": [
    {
      "DeleteOnTermination": true,
      "DeviceIndex": 0,
      "SubnetId": "subnet-049df61146c4d7901"
    }
  ],
  "Placement": {
    "AvailabilityZone": "us-east-2b",
    "Tenancy": "default"
  },
  "Monitoring": {
    "Enabled": false
  }
},
"LaunchedAvailabilityZone": "us-east-2b",
"ProductDescription": "Linux/UNIX",
"SpotInstanceRequestId": "sir-08b93456",
"SpotPrice": "0.010000"
"State": "active",
"Status": {
  "Code": "fulfilled",
```

```

        "Message": "Your Spot request is fulfilled.",
        "UpdateTime": "2018-04-30T18:16:21.000Z"
    },
    "Tags": [],
    "Type": "one-time",
    "InstanceInterruptionBehavior": "terminate"
}
]
}

```

Exemple 2 : pour décrire les demandes d'instance Spot basées sur des filtres

L'`describe-spot-instance-requests` exemple suivant utilise des filtres pour étendre les résultats aux demandes d'instance Spot avec le type d'instance spécifié dans la zone de disponibilité spécifiée. L'exemple utilise le `--query` paramètre pour afficher uniquement les ID d'instance.

```

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

```

Sortie :

```

i-057750d42936e468a
i-001efd250faaa6ffa
i-027552a73f021f3bd
...

```

Pour d'autres exemples d'utilisation de filtres, consultez la section [Répertoire et filtrer vos ressources](#) dans le guide de l'utilisateur d'Amazon Elastic Compute Cloud.

Exemple 3 : pour décrire les demandes d'instance Spot basées sur des balises

L'`describe-spot-instance-requests` exemple suivant utilise des filtres de balises pour étendre les résultats aux demandes d'instance Spot dotées de la balise `cost-center=cc123`.

```

aws ec2 describe-spot-instance-requests \
  --filters Name=tag:cost-center,Values=cc123

```

Pour obtenir un exemple de la manière de procéder pour `describe-spot-instance-requests`, reportez-vous à l'exemple 1.

Pour d'autres exemples d'utilisation de filtres de balises, veuillez consulter la rubrique [Utiliser des balises](#) dans le Guide de l'utilisateur Amazon EC2.

- Pour plus de détails sur l'API, reportez-vous [DescribeSpotInstanceRequests](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit la demande d'instance Spot spécifiée.

```
Get-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

Sortie :

```
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
```

Exemple 2 : Cet exemple décrit toutes vos demandes d'instance Spot.

```
Get-EC2SpotInstanceRequest
```

- Pour plus de détails sur l'API, reportez-vous [DescribeSpotInstanceRequests](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeSpotPriceHistory** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeSpotPriceHistory`.

CLI

AWS CLI

Pour décrire l'historique des prix au comptant

Cet exemple de commande renvoie l'historique des prix au comptant pour les instances `m1.xlarge` pour un jour donné de janvier.

Commande :

```
aws ec2 describe-spot-price-history --instance-types m1.xlarge --start-time 2014-01-06T07:08:09 --end-time 2014-01-06T08:09:10
```

Sortie :

```
{
  "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"
    },
    ...
}
```

Pour décrire l'historique des prix au comptant pour Linux/UNIX, Amazon VPC

Cet exemple de commande renvoie l'historique des prix au comptant pour les instances Amazon VPC m1.xlarge, Linux/UNIX pour un jour donné de janvier.

Commande :

```
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
```

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [DescribeSpotPriceHistory](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple obtient les 10 dernières entrées de l'historique des prix au comptant pour le type d'instance et la zone de disponibilité spécifiés. Notez que la valeur spécifiée pour le AvailabilityZone paramètre - doit être valide pour la valeur de région fournie au paramètre -Region de l'applet de commande (non illustré dans l'exemple) ou définie par défaut dans le shell. Cet exemple de commande suppose qu'une région par défaut « us-west-2 » a été définie dans l'environnement.

```
Get-EC2SpotPriceHistory -InstanceType c3.large -AvailabilityZone us-west-2a -
MaxResult 10
```

Sortie :

```
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
...
```

- Pour plus de détails sur l'API, reportez-vous [DescribeSpotPriceHistory](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeSubnets** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeSubnets`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Créer et gérer un service résilient](#)

.NET

AWS SDK for .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Get all the subnets for a Vpc in a set of availability zones.
/// </summary>
/// <param name="vpcId">The Id of the Vpc.</param>
/// <param name="availabilityZones">The list of availability zones.</param>
/// <returns>The collection of subnet objects.</returns>
public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
List<string> availabilityZones)
{
    var subnets = new List<Subnet>();
    var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
        new DescribeSubnetsRequest()
        {
            Filters = new List<Amazon.EC2.Model.Filter>()
            {
                new ("vpc-id", new List<string>() { vpcId}),
                new ("availability-zone", availabilityZones),
                new ("default-for-az", new List<string>() { "true" })
            }
        });
}
```

```
// Get the entire list using the paginator.
await foreach (var subnet in subnetPaginator.Subnets)
{
    subnets.Add(subnet);
}

return subnets;
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeSubnets](#) à la section Référence des AWS SDK for .NET API.

CLI

AWS CLI

Exemple 1 : description de l'ensemble de vos sous-réseaux

L'exemple `describe-subnets` suivant affiche les détails de vos sous-réseaux.

```
aws ec2 describe-subnets
```

Sortie :

```
{
  "Subnets": [
    {
      "AvailabilityZone": "us-east-1d",
      "AvailabilityZoneId": "use1-az2",
      "AvailableIpAddressCount": 4089,
      "CidrBlock": "172.31.80.0/20",
      "DefaultForAz": true,
      "MapPublicIpOnLaunch": false,
      "MapCustomerOwnedIpOnLaunch": true,
      "State": "available",
      "SubnetId": "subnet-0bb1c79de3EXAMPLE",
      "VpcId": "vpc-0ee975135dEXAMPLE",
      "OwnerId": "111122223333",
      "AssignIpv6AddressOnCreation": false,
      "Ipv6CidrBlockAssociationSet": [],
      "CustomerOwnedIpv4Pool": "pool-2EXAMPLE",
    }
  ]
}
```

```

        "SubnetArn": "arn:aws:ec2:us-east-2:111122223333:subnet/
subnet-0bb1c79de3EXAMPLE",
        "EnableDns64": false,
        "Ipv6Native": false,
        "PrivateDnsNameOptionsOnLaunch": {
            "HostnameType": "ip-name",
            "EnableResourceNameDnsARecord": false,
            "EnableResourceNameDnsAAAARecord": false
        }
    },
    {
        "AvailabilityZone": "us-east-1d",
        "AvailabilityZoneId": "use1-az2",
        "AvailableIpAddressCount": 4089,
        "CidrBlock": "172.31.80.0/20",
        "DefaultForAz": true,
        "MapPublicIpOnLaunch": true,
        "MapCustomerOwnedIpOnLaunch": false,
        "State": "available",
        "SubnetId": "subnet-8EXAMPLE",
        "VpcId": "vpc-3EXAMPLE",
        "OwnerId": "111122223333",
        "AssignIpv6AddressOnCreation": false,
        "Ipv6CidrBlockAssociationSet": [],
        "Tags": [
            {
                "Key": "Name",
                "Value": "MySubnet"
            }
        ],
        "SubnetArn": "arn:aws:ec2:us-east-1:111122223333:subnet/
subnet-8EXAMPLE",
        "EnableDns64": false,
        "Ipv6Native": false,
        "PrivateDnsNameOptionsOnLaunch": {
            "HostnameType": "ip-name",
            "EnableResourceNameDnsARecord": false,
            "EnableResourceNameDnsAAAARecord": false
        }
    }
]
}

```

Pour plus d'informations, veuillez consulter la rubrique [Working with VPCs and Subnets](#) dans le Guide de l'utilisateur AWS .

Exemple 2 : description des sous-réseaux d'un VPC spécifique

L'exemple `describe-subnets` suivant utilise un filtre pour récupérer les détails des sous-réseaux du VPC spécifié.

```
aws ec2 describe-subnets \  
  --filters "Name=vpc-id,Values=vpc-3EXAMPLE"
```

Sortie :

```
{  
  "Subnets": [  
    {  
      "AvailabilityZone": "us-east-1d",  
      "AvailabilityZoneId": "use1-az2",  
      "AvailableIpAddressCount": 4089,  
      "CidrBlock": "172.31.80.0/20",  
      "DefaultForAz": true,  
      "MapPublicIpOnLaunch": true,  
      "MapCustomerOwnedIpOnLaunch": false,  
      "State": "available",  
      "SubnetId": "subnet-8EXAMPLE",  
      "VpcId": "vpc-3EXAMPLE",  
      "OwnerId": "1111222233333",  
      "AssignIpv6AddressOnCreation": false,  
      "Ipv6CidrBlockAssociationSet": [],  
      "Tags": [  
        {  
          "Key": "Name",  
          "Value": "MySubnet"  
        }  
      ],  
      "SubnetArn": "arn:aws:ec2:us-east-1:111122223333:subnet/  
subnet-8EXAMPLE",  
      "EnableDns64": false,  
      "Ipv6Native": false,  
      "PrivateDnsNameOptionsOnLaunch": {  
        "HostnameType": "ip-name",  
        "EnableResourceNameDnsARecord": false,  
        "EnableResourceNameDnsAAAARecord": false
```

```
    }  
  }  
]  
}
```

Pour plus d'informations, veuillez consulter la rubrique [Working with VPCs and Subnets](#) dans le Guide de l'utilisateur AWS .

Exemple 3 : description des sous-réseaux avec une balise spécifique

L'exemple `describe-subnets` suivant utilise un filtre pour récupérer les détails des sous-réseaux comportant la balise `CostCenter=123` et le paramètre `--query` permettant d'afficher les ID des sous-réseaux comportant cette balise.

```
aws ec2 describe-subnets \  
  --filters "Name=tag:CostCenter,Values=123" \  
  --query "Subnets[*].SubnetId" \  
  --output text
```

Sortie :

```
subnet-0987a87c8b37348ef  
subnet-02a95061c45f372ee  
subnet-03f720e7de2788d73
```

Pour de plus amples informations, veuillez consulter la rubrique [Working with VPCs and Subnets](#) dans le Guide de l'utilisateur Amazon VPC.

- Pour plus de détails sur l'API, reportez-vous [DescribeSubnets](#) à la section Référence des AWS CLI commandes.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
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"] },
    ],
  }),
);
```

- Pour plus de détails sur l'API, reportez-vous [DescribeSubnets](#) à la section Référence des AWS SDK for JavaScript API.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit le sous-réseau spécifié.

```
Get-EC2Subnet -SubnetId subnet-1a2b3c4d
```

Sortie :

```
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
```

Exemple 2 : Cet exemple décrit tous vos sous-réseaux.

```
Get-EC2Subnet
```

- Pour plus de détails sur l'API, reportez-vous [DescribeSubnets](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Python

SDK pour Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

    def __init__(
        self,
        resource_prefix,
        inst_type,
        ami_param,
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    ):
        """
        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is
                created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
```

```
self.iam_client = iam_client
self.launch_template_name = f"{resource_prefix}-template"
self.group_name = f"{resource_prefix}-group"
self.instance_policy_name = f"{resource_prefix}-pol"
self.instance_role_name = f"{resource_prefix}-role"
self.instance_profile_name = f"{resource_prefix}-prof"
self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
self.bad_creds_role_name = f"{resource_prefix}-bc-role"
self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"
self.key_pair_name = f"{resource_prefix}-key-pair"

def get_subnets(self, vpc_id, zones):
    """
    Gets the default subnets in a VPC for a specified list of Availability
    Zones.

    :param vpc_id: The ID of the VPC to look up.
    :param zones: The list of Availability Zones to look up.
    :return: The list of subnets found.
    """
    try:
        response = self.ec2_client.describe_subnets(
            Filters=[
                {"Name": "vpc-id", "Values": [vpc_id]},
                {"Name": "availability-zone", "Values": zones},
                {"Name": "default-for-az", "Values": ["true"]},
            ]
        )
        subnets = response["Subnets"]
        log.info("Found %s subnets for the specified zones.", len(subnets))
    except ClientError as err:
        raise AutoScalerError(f"Couldn't get subnets: {err}")
    else:
        return subnets
```

- Pour plus de détails sur l'API, consultez [DescribeSubnets](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeTags** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeTags`.

CLI

AWS CLI

Exemple 1 : pour décrire toutes les balises d'une seule ressource

L'`describe-tagsexemple` suivant décrit les balises de l'instance spécifiée.

```
aws ec2 describe-tags \  
  --filters "Name=resource-id,Values=i-1234567890abcdef8"
```

Sortie :

```
{  
  "Tags": [  
    {  
      "ResourceType": "instance",  
      "ResourceId": "i-1234567890abcdef8",  
      "Value": "Test",  
      "Key": "Stack"  
    },  
    {  
      "ResourceType": "instance",  
      "ResourceId": "i-1234567890abcdef8",  
      "Value": "Beta Server",  
      "Key": "Name"  
    }  
  ]  
}
```

Exemple 2 : pour décrire toutes les balises d'un type de ressource

L'`describe-tagsexemple` suivant décrit les balises de vos volumes.

```
aws ec2 describe-tags \  
  --filters "Name=resource-id,Values=i-1234567890abcdef8"
```

```
--filters "Name=resource-type,Values=volume"
```

Sortie :

```
{
  "Tags": [
    {
      "ResourceType": "volume",
      "ResourceId": "vol-1234567890abcdef0",
      "Value": "Project1",
      "Key": "Purpose"
    },
    {
      "ResourceType": "volume",
      "ResourceId": "vol-049df61146c4d7901",
      "Value": "Logs",
      "Key": "Purpose"
    }
  ]
}
```

Exemple 3 : Pour décrire tous vos tags

L'`describe-tagsexemple` suivant décrit les balises de toutes vos ressources.

```
aws ec2 describe-tags
```

Exemple 4 : Pour décrire les balises de vos ressources en fonction d'une clé de balise

L'`describe-tagsexemple` suivant décrit les balises de vos ressources dotées d'une balise avec la clé `Stack`.

```
aws ec2 describe-tags \
  --filters Name=key,Values=Stack
```

Sortie :

```
{
  "Tags": [
    {
      "ResourceType": "volume",
      "ResourceId": "vol-027552a73f021f3b",
```

```

        "Value": "Production",
        "Key": "Stack"
    },
    {
        "ResourceType": "instance",
        "ResourceId": "i-1234567890abcdef8",
        "Value": "Test",
        "Key": "Stack"
    }
]
}

```

Exemple 5 : pour décrire les balises de vos ressources en fonction d'une clé de balise et d'une valeur de balise

L'`describe-tags`exemple suivant décrit les balises associées à vos ressources `Stack=Test`.

```

aws ec2 describe-tags \
  --filters Name=key,Values=Stack Name=value,Values=Test

```

Sortie :

```

{
  "Tags": [
    {
      "ResourceType": "image",
      "ResourceId": "ami-3ac336533f021f3bd",
      "Value": "Test",
      "Key": "Stack"
    },
    {
      "ResourceType": "instance",
      "ResourceId": "i-1234567890abcdef8",
      "Value": "Test",
      "Key": "Stack"
    }
  ]
}

```

L'`describe-tags`exemple suivant utilise une syntaxe alternative pour décrire les ressources associées à la balise `Stack=Test`.

```
aws ec2 describe-tags \
  --filters "Name=tag:Stack,Values=Test"
```

L'exemple de commande `describe-tags` suivant décrit les balises de toutes vos instances dont une balise comporte une clé `Purpose` mais aucune valeur.

```
aws ec2 describe-tags \
  --filters "Name=resource-type,Values=instance" "Name=key,Values=Purpose"
  "Name=value,Values="
```

Sortie :

```
{
  "Tags": [
    {
      "ResourceType": "instance",
      "ResourceId": "i-1234567890abcdef5",
      "Value": null,
      "Key": "Purpose"
    }
  ]
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeTags](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple récupère les balises pour le type de ressource « image »

```
Get-EC2Tag -Filter @{Name="resource-type";Values="image"}
```

Sortie :

Key	ResourceId	ResourceType	Value
---	-----	-----	-----
Name	ami-0a123b4ccb567a8ea	image	Win7-Imported

```
auto-delete ami-0a123b4ccb567a8ea image never
```

Exemple 2 : Cet exemple récupère toutes les balises de toutes les ressources et les regroupe par type de ressource

```
Get-EC2Tag | Group-Object resourcetype
```

Sortie :

```
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}
```

Exemple 3 : Cet exemple affiche toutes les ressources avec le tag « auto-delete » avec la valeur « non » pour la région donnée

```
Get-EC2Tag -Region eu-west-1 -Filter @{Name="tag:auto-delete";Values="no"}
```

Sortie :

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

Exemple 4 : Cet exemple obtient toutes les ressources avec la balise « auto-delete » avec une valeur « aucune » et des filtres supplémentaires dans le canal suivant pour analyser uniquement les types de ressources « instance » et crée finalement la balise « ThisInstance » pour chaque ressource d'instance, la valeur étant l'identifiant de l'instance lui-même

```
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}}
```

Exemple 5 : Cet exemple récupère les balises pour toutes les ressources de l'instance ainsi que les clés « Nom » et les affiche sous forme de tableau

```
Get-EC2Tag -Filter @{"Name"="resource-
type";Values="instance"},@{"Name"="key";Values="Name"} | Select-Object ResourceId,
@{"Name"="Name-Tag";Expression={$PSItem.Value}} | Format-Table -AutoSize
```

Sortie :

ResourceId	Name-Tag
-----	-----
i-012e3cb4df567e1aa	jump1
i-01c23a45d6fc7a89f	repro-3

- Pour plus de détails sur l'API, reportez-vous [DescribeTags](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation `DescribeVolumeAttribute` avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeVolumeAttribute`.

CLI

AWS CLI

Pour décrire un attribut de volume

Cet exemple de commande décrit l'attribut `autoEnableIO` du volume avec l'ID `vol-049df61146c4d7901`.

Commande :

```
aws ec2 describe-volume-attribute --volume-id vol-049df61146c4d7901 --attribute autoEnableIO
```

Sortie :

```
{
  "AutoEnableIO": {
    "Value": false
  },
  "VolumeId": "vol-049df61146c4d7901"
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeVolumeAttribute](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit l'attribut spécifié du volume spécifié.

```
Get-EC2VolumeAttribute -VolumeId vol-12345678 -Attribute AutoEnableIO
```

Sortie :

```
AutoEnableIO    ProductCodes    VolumeId
```

```
-----      -----      -----  
False        {}          vol-12345678
```

- Pour plus de détails sur l'API, reportez-vous [DescribeVolumeAttribute](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeVolumeStatus** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeVolumeStatus`.

CLI

AWS CLI

Pour décrire l'état d'un volume unique

Cet exemple de commande décrit l'état du volume `vol-1234567890abcdef0`.

Commande :

```
aws ec2 describe-volume-status --volume-ids vol-1234567890abcdef0
```

Sortie :

```
{  
  "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": []
    }
  ]
}
```

Pour décrire l'état des volumes altérés

Cet exemple de commande décrit l'état de tous les volumes endommagés. Dans cet exemple de sortie, aucun volume n'est altéré.

Commande :

```
aws ec2 describe-volume-status --filters Name=volume-
status.status,Values=impaired
```

Sortie :

```
{
  "VolumeStatuses": []
}
```

Si la vérification de l'état d'un volume a échoué (l'état est altéré), consultez la section Utilisation d'un volume altéré dans le guide de l'utilisateur Amazon EC2.

- Pour plus de détails sur l'API, reportez-vous [DescribeVolumeStatus](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit l'état du volume spécifié.

```
Get-EC2VolumeStatus -VolumeId vol-12345678
```

Sortie :

```

Actions           : {}
AvailabilityZone  : us-west-2a
Events           : {}
VolumeId         : vol-12345678
VolumeStatus     : Amazon.EC2.Model.VolumeStatusInfo

```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus
```

Sortie :

Details	Status
-----	-----
{io-enabled, io-performance}	ok

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus.Details
```

Sortie :

Name	Status
----	-----
io-enabled	passed
io-performance	not-applicable

- Pour plus de détails sur l'API, reportez-vous [DescribeVolumeStatus](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeVolumes** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeVolumes`.

CLI

AWS CLI

Exemple 1 : pour décrire un volume

L'`describe-volumes` suivant décrit les volumes spécifiés dans la région actuelle.

```
aws ec2 describe-volumes \  
  --volume-ids vol-049df61146c4d7901 vol-1234567890abcdef0
```

Sortie :

```
{  
  "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  
    }  
  ]  
}
```

```
    }  
  ]  
}
```

Exemple 2 : pour décrire les volumes attachés à une instance spécifique

L'`describe-volumes` exemple suivant décrit tous les volumes qui sont à la fois attachés à l'instance spécifiée et définis pour être supprimés lorsque l'instance se termine.

```
aws ec2 describe-volumes \  
  --region us-east-1 \  
  --filters Name=attachment.instance-id,Values=i-1234567890abcdef0  
  Name=attachment.delete-on-termination,Values=true
```

Pour obtenir un exemple de la manière de procéder pour `describe-volumes`, reportez-vous à l'exemple 1.

Exemple 3 : pour décrire les volumes disponibles dans une zone de disponibilité spécifique

L'`describe-volumes` exemple suivant décrit tous les volumes dont le statut est `available` et se trouvent dans la zone de disponibilité spécifiée.

```
aws ec2 describe-volumes \  
  --filters Name=status,Values=available Name=availability-zone,Values=us-  
east-1a
```

Pour obtenir un exemple de la manière de procéder pour `describe-volumes`, reportez-vous à l'exemple 1.

Exemple 4 : pour décrire des volumes en fonction de balises

L'`describe-volumes` exemple suivant décrit tous les volumes dotés de la clé de balise `Name` et d'une valeur commençant par `Test`. La sortie est ensuite filtrée avec une requête qui affiche uniquement les balises et les identifiants des volumes.

```
aws ec2 describe-volumes \  
  --filters Name=tag:Name,Values=Test* \  
  --query "Volumes[*].{ID:VolumeId,Tag:Tags}"
```

Sortie :

```
[
```

```
[
  {
    "Tag": [
      {
        "Value": "Test2",
        "Key": "Name"
      }
    ],
    "ID": "vol-1234567890abcdef0"
  },
  {
    "Tag": [
      {
        "Value": "Test1",
        "Key": "Name"
      }
    ],
    "ID": "vol-049df61146c4d7901"
  }
]
```

Pour d'autres exemples d'utilisation de filtres de balises, veuillez consulter la rubrique [Utiliser des balises](#) dans le Guide de l'utilisateur Amazon EC2.

- Pour plus de détails sur l'API, reportez-vous [DescribeVolumes](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit le volume EBS spécifié.

```
Get-EC2Volume -VolumeId vol-12345678
```

Sortie :

```
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
```

Exemple 2 : Cet exemple décrit vos volumes EBS dont le statut est « disponible ».

```
Get-EC2Volume -Filter @{ Name="status"; Values="available" }
```

Sortie :

```
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
...
```

Exemple 3 : Cet exemple décrit tous vos volumes EBS.

```
Get-EC2Volume
```

- Pour plus de détails sur l'API, reportez-vous [DescribeVolumes](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeVpcAttribute** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeVpcAttribute`.

CLI

AWS CLI

Pour décrire l' `enableDnsSupport` attribut

Cet exemple décrit l'`enableDnsSupport` attribut. Cet attribut indique si la résolution DNS est activée pour le VPC. Si cet attribut est `true`, le serveur Amazon DNS résout les noms d'hôte DNS de vos instances par rapport aux adresses IP correspondantes. Dans le cas contraire, il ne le fait pas.

Commande :

```
aws ec2 describe-vpc-attribute --vpc-id vpc-a01106c2 --attribute enableDnsSupport
```

Sortie :

```
{
  "VpcId": "vpc-a01106c2",
  "EnableDnsSupport": {
    "Value": true
  }
}
```

Pour décrire l' `enableDnsHostnames` attribut

Cet exemple décrit l'`enableDnsHostnames` attribut. Cet attribut indique si les instances lancées dans le VPC obtiennent des noms d'hôte DNS. Si l'attribut est défini sur `true`, les instances incluses dans le VPC ont obtenu des noms d'hôte DNS.

Commande :

```
aws ec2 describe-vpc-attribute --vpc-id vpc-a01106c2 --attribute
enableDnsHostnames
```

Sortie :

```
{
  "VpcId": "vpc-a01106c2",
```

```
"EnableDnsHostnames": {  
  "Value": true  
}  
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeVpcAttribute](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit l'attribut enableDnsSupport « ».

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsSupport
```

Sortie :

```
EnableDnsSupport  
-----  
True
```

Exemple 2 : Cet exemple décrit l'attribut enableDnsHostnames « ».

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsHostnames
```

Sortie :

```
EnableDnsHostnames  
-----  
True
```

- Pour plus de détails sur l'API, reportez-vous [DescribeVpcAttribute](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeVpcClassicLink** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeVpcClassicLink`.

CLI

AWS CLI

Pour décrire le ClassicLink statut de vos VPC

Cet exemple répertorie le ClassicLink statut de `vpc-88888888`.

Commande :

```
aws ec2 describe-vpc-classic-link --vpc-id vpc-88888888
```

Sortie :

```
{
  "Vpcs": [
    {
      "ClassicLinkEnabled": true,
      "VpcId": "vpc-88888888",
      "Tags": [
        {
          "Value": "classiclinkvpc",
          "Key": "Name"
        }
      ]
    }
  ]
}
```

Cet exemple répertorie uniquement les VPC activés pour Classiclink (la valeur du filtre `is-classic-link-enabled` est définie sur `true`)

Commande :

```
aws ec2 describe-vpc-classic-link --filter "Name=is-classic-link-enabled,Values=true"
```

- Pour plus de détails sur l'API, reportez-vous [DescribeVpcClassicLink](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : L'exemple ci-dessus renvoie tous les VPC avec leur ClassicLinkEnabled état pour la région

```
Get-EC2VpcClassicLink -Region eu-west-1
```

Sortie :

```
ClassicLinkEnabled Tags    VpcId
-----
False              {Name} vpc-0fc1ff23f45b678eb
False              {}      vpc-01e23c4a5d6db78e9
False              {Name} vpc-0123456b078b9d01f
False              {}      vpc-12cf3b4f
False              {Name} vpc-0b12d3456a7e8901d
```

- Pour plus de détails sur l'API, reportez-vous [DescribeVpcClassicLink](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeVpcClassicLinkDnsSupport** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeVpcClassicLinkDnsSupport`.

CLI

AWS CLI

Pour décrire la prise en charge du ClassicLink DNS pour vos VPC

Cet exemple décrit l'état de support ClassicLink DNS de tous vos VPC.

Commande :

```
aws ec2 describe-vpc-classic-link-dns-support
```

Sortie :

```
{
  "Vpcs": [
    {
      "VpcId": "vpc-88888888",
      "ClassicLinkDnsSupported": true
    },
    {
      "VpcId": "vpc-1a2b3c4d",
      "ClassicLinkDnsSupported": false
    }
  ]
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeVpcClassicLinkDnsSupport](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit le statut de support ClassicLink DNS des VPC pour la région eu-west-1

```
Get-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

Sortie :

```
ClassicLinkDnsSupported VpcId
-----
False                   vpc-0b12d3456a7e8910d
False                   vpc-12cf3b4f
```

- Pour plus de détails sur l'API, reportez-vous [DescribeVpcClassicLinkDnsSupport](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeVpcEndpointServices** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeVpcEndpointServices`.

CLI

AWS CLI

Exemple 1 : pour décrire tous les services de point de terminaison VPC

L'exemple « `describe-vpc-endpoint-services` » suivant répertorie tous les services de point de terminaison VPC pour une AWS région.

```
aws ec2 describe-vpc-endpoint-services
```

Sortie :

```
{
  "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"
]
}

```

Pour plus d'informations, voir [Afficher les noms des AWS services disponibles](#) dans le guide de l'utilisateur pour AWS PrivateLink.

Exemple 2 : pour décrire les détails d'un service de point de terminaison

L'exemple « describe-vpc-endpoint-services » suivant répertorie les détails du service de point de terminaison de l'interface Amazon S3

```

aws ec2 describe-vpc-endpoint-services \
  --filter "Name=service-type,Values=Interface" Name=service-
name,Values=com.amazonaws.us-east-1.s3

```

Sortie :

```

{
  "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"
]
}

```

Pour plus d'informations, voir [Afficher les noms des AWS services disponibles](#) dans le guide de l'utilisateur pour AWS PrivateLink.

- Pour plus de détails sur l'API, reportez-vous [DescribeVpcEndpointServices](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit le service de point de terminaison VPC EC2 avec le filtre donné, dans ce cas `com.amazonaws.eu-west-1.ecs`. En outre, il agrandit également la `ServiceDetails` propriété et affiche les détails

```

Get-EC2VpcEndpointService -Region eu-west-1 -MaxResult 5 -Filter @{Name="service-name";Values="com.amazonaws.eu-west-1.ecs"} | Select-Object -ExpandProperty ServiceDetails

```

Sortie :

```

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

```

Exemple 2 : Cet exemple récupère tous les services de point de terminaison VPC EC2 et renvoie le ServiceNames « ssm » correspondant

```

Get-EC2VpcEndpointService -Region eu-west-1 | Select-Object -ExpandProperty
  Servicenames | Where-Object { -match "ssm"}

```

Sortie :

```

com.amazonaws.eu-west-1.ssm
com.amazonaws.eu-west-1.ssmmessages

```

- Pour plus de détails sur l'API, reportez-vous [DescribeVpcEndpointServices](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeVpcEndpoints** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeVpcEndpoints`.

CLI**AWS CLI**

Pour décrire vos points de terminaison VPC

L'exemple suivant affiche les détails de tous vos points de terminaison VPC.

```
aws ec2 describe-vpc-endpoints
```

Sortie :

```
{
  "VpcEndpoints": [
    {
      "PolicyDocument": "{\"Version\":\"2008-10-17\",\"Statement\":
[{\n  \"Effect\": \"Allow\", \n  \"Principal\": \"*\", \n  \"Action\": \"*\", \n  \"Resource\": \"*\"
}]}\",
      "VpcId": "vpc-aabb1122",
      "NetworkInterfaceIds": [],
      "SubnetIds": [],
      "PrivateDnsEnabled": true,
      "State": "available",
      "ServiceName": "com.amazonaws.us-east-1.dynamodb",
      "RouteTableIds": [
        "rtb-3d560345"
      ],
      "Groups": [],
      "VpcEndpointId": "vpce-032a826a",
      "VpcEndpointType": "Gateway",
      "CreationTimestamp": "2017-09-05T20:41:28Z",
      "DnsEntries": [],
      "OwnerId": "123456789012"
    },
    {
      "PolicyDocument": "{\n  \"Statement\": [\n    {\n      \"Action\":
\"*\", \n      \"Effect\": \"Allow\", \n      \"Principal\": \"*\", \n
      \"Resource\": \"*\" \n    } \n  ]\n}",
      "VpcId": "vpc-1a2b3c4d",
      "NetworkInterfaceIds": [
        "eni-2ec2b084",
        "eni-1b4a65cf"
      ],
      "SubnetIds": [
        "subnet-d6fcaa8d",
        "subnet-7b16de0c"
      ],
      "PrivateDnsEnabled": false,
      "State": "available",
      "ServiceName": "com.amazonaws.us-east-1.elasticloadbalancing",
      "RouteTableIds": [],
    }
  ]
}
```

```

    "Groups": [
      {
        "GroupName": "default",
        "GroupId": "sg-54e8bf31"
      }
    ],
    "VpcEndpointId": "vpce-0f89a33420c1931d7",
    "VpcEndpointType": "Interface",
    "CreationTimestamp": "2017-09-05T17:55:27.583Z",
    "DnsEntries": [
      {
        "HostedZoneId": "Z7HUB22UULQXV",
        "DnsName": "vpce-0f89a33420c1931d7-
bluzidnv.elasticloadbalancing.us-east-1.vpce.amazonaws.com"
      },
      {
        "HostedZoneId": "Z7HUB22UULQXV",
        "DnsName": "vpce-0f89a33420c1931d7-bluzidnv-us-
east-1b.elasticloadbalancing.us-east-1.vpce.amazonaws.com"
      },
      {
        "HostedZoneId": "Z7HUB22UULQXV",
        "DnsName": "vpce-0f89a33420c1931d7-bluzidnv-us-
east-1a.elasticloadbalancing.us-east-1.vpce.amazonaws.com"
      }
    ],
    "OwnerId": "123456789012"
  },
  {
    "VpcEndpointId": "vpce-aabbaabbaabbaabba",
    "VpcEndpointType": "GatewayLoadBalancer",
    "VpcId": "vpc-111122223333aabbc",
    "ServiceName": "com.amazonaws.vpce.us-east-1.vpce-
svc-123123a1c43abc123",
    "State": "available",
    "SubnetIds": [
      "subnet-0011aabbcc2233445"
    ],
    "RequesterManaged": false,
    "NetworkInterfaceIds": [
      "eni-01010120203030405"
    ],
    "CreationTimestamp": "2020-11-11T08:06:03.522Z",
    "Tags": [],
  }

```

```

        "OwnerId": "123456789012"
    }
]
}

```

Pour de plus amples informations, consultez [Points de terminaison VPC](#) dans le Guide de l'utilisateur Amazon VPC.

- Pour plus de détails sur l'API, reportez-vous [DescribeVpcEndpoints](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit un ou plusieurs de vos points de terminaison VPC pour la région eu-west-1. Il dirige ensuite la sortie vers la commande suivante, qui sélectionne la `VpcEndpointId` propriété et renvoie l'ID VPC du tableau sous forme de tableau de chaînes

```

Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object -ExpandProperty
VpcEndpointId

```

Sortie :

```

vpce-01a2ab3f4f5cc6f7d
vpce-01d2b345a6787890b
vpce-0012e34d567890e12
vpce-0c123db4567890123

```

Exemple 2 : Cet exemple décrit tous les points de terminaison VPC pour la région eu-west-1 et sélectionne `VpcEndpointId`, `VpcId` ainsi que les propriétés pour les présenter sous forme de `ServiceName` tableau `PrivateDnsEnabled`

```

Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object VpcEndpointId, VpcId,
ServiceName, PrivateDnsEnabled | Format-Table -AutoSize

```

Sortie :

VpcEndpointId	VpcId	ServiceName
PrivateDnsEnabled		

```

-----
-----
-----
vpce-02a2ab2f2f2cc2f2d vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ssm
    True
vpce-01d1b111a1114561b vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ec2
    True
vpce-0011e23d45167e838 vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ec2messages
    True
vpce-0c123db4567890123 vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ssmessages
    True

```

Exemple 3 : Cet exemple exporte le document de politique pour le point de terminaison VPC vpce-01a2ab3f4f5cc6f7d dans un fichier json

```

Get-EC2VpcEndpoint -Region eu-west-1 -VpcEndpointId vpce-01a2ab3f4f5cc6f7d |
  Select-Object -expand PolicyDocument | Out-File vpce_policyDocument.json

```

- Pour plus de détails sur l'API, reportez-vous [DescribeVpcEndpoints](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeVpcs** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeVpcs`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Créer et gérer un service résilient](#)

.NET

AWS SDK for .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Get the default VPC for the account.
/// </summary>
/// <returns>The default VPC object.</returns>
public async Task<Vpc> GetDefaultVpc()
{
    var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
        new DescribeVpcsRequest()
        {
            Filters = new List<Amazon.EC2.Model.Filter>()
            {
                new ("is-default", new List<string>() { "true" })
            }
        });
    return vpcResponse.Vpcs[0];
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeVpcs](#) à la section Référence des AWS SDK for .NET API.

CLI

AWS CLI

Exemple 1 : description de l'ensemble de vos VPC

L'exemple `describe-vpcs` suivant récupère les détails de vos VPC.

```
aws ec2 describe-vpcs
```

Sortie :

```
{
  "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"
            }
        ]
    }
]
}

```

Exemple 2 : description d'un VPC spécifié

L'exemple `describe-vpcs` suivant récupère les détails du VPC spécifié.

```

aws ec2 describe-vpcs \
  --vpc-ids vpc-06e4ab6c6cEXAMPLE

```

Sortie :

```

{
  "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"
        }
      ]
    }
  ]
}

```

```
}
  ]
}
]
```

- Pour plus de détails sur l'API, reportez-vous [DescribeVpcs](#) à la section Référence des AWS CLI commandes.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
const client = new EC2Client({});
const { Vpcs } = await client.send(
  new DescribeVpcsCommand({
    Filters: [{ Name: "is-default", Values: ["true"] }],
  }),
);
```

- Pour plus de détails sur l'API, reportez-vous [DescribeVpcs](#) à la section Référence des AWS SDK for JavaScript API.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit le VPC spécifié.

```
Get-EC2Vpc -VpcId vpc-12345678
```

Sortie :

```
CidrBlock      : 10.0.0.0/16
DhcpOptionsId  : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault      : False
State          : available
Tags           : {Name}
VpcId          : vpc-12345678
```

Exemple 2 : Cet exemple décrit le VPC par défaut (il ne peut y en avoir qu'un par région). Si votre compte prend en charge EC2-Classique dans cette région, il n'existe pas de VPC par défaut.

```
Get-EC2Vpc -Filter @{"Name"="isDefault"; Values="true"}
```

Sortie :

```
CidrBlock      : 172.31.0.0/16
DhcpOptionsId  : dopt-12345678
InstanceTenancy : default
IsDefault      : True
State          : available
Tags           : {}
VpcId          : vpc-45678901
```

Exemple 3 : Cet exemple décrit les VPC qui correspondent au filtre spécifié (c'est-à-dire qui ont un CIDR correspondant à la valeur « 10.0.0.0/16 » et dont l'état est « disponible »).

```
Get-EC2Vpc -Filter @{"Name"="cidr";
  Values="10.0.0.0/16"},@{"Name"="state";Values="available"}
```

Exemple 4 : Cet exemple décrit tous vos VPC.

```
Get-EC2Vpc
```

- Pour plus de détails sur l'API, reportez-vous [DescribeVpcs](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Python

SDK pour Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

    def __init__(
        self,
        resource_prefix,
        inst_type,
        ami_param,
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    ):
        """
        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is
                created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
```

```
self.iam_client = iam_client
self.launch_template_name = f"{resource_prefix}-template"
self.group_name = f"{resource_prefix}-group"
self.instance_policy_name = f"{resource_prefix}-pol"
self.instance_role_name = f"{resource_prefix}-role"
self.instance_profile_name = f"{resource_prefix}-prof"
self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
self.bad_creds_role_name = f"{resource_prefix}-bc-role"
self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"
self.key_pair_name = f"{resource_prefix}-key-pair"

def get_default_vpc(self):
    """
    Gets the default VPC for the account.

    :return: Data about the default VPC.
    """
    try:
        response = self.ec2_client.describe_vpcs(
            Filters=[{"Name": "is-default", "Values": ["true"]}])
    except ClientError as err:
        raise AutoScalerError(f"Couldn't get default VPC: {err}")
    else:
        return response["Vpcs"][0]
```

- Pour plus de détails sur l'API, consultez [DescribeVpcs](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeVpnConnections** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeVpnConnections`.

CLI

AWS CLI

Exemple 1 : Pour décrire vos connexions VPN

L'`describe-vpn-connection`exemple suivant décrit toutes vos connexions VPN de site à site.

```
aws ec2 describe-vpn-connections
```

Sortie :

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

Pour plus d'informations, consultez [Comment fonctionne le VPN de AWS site à site dans le guide de l'utilisateur du VPN](#) de site à site.AWS

Exemple 2 : Pour décrire vos connexions VPN disponibles

L'`describe-vpn-connections`exemple suivant décrit vos connexions VPN de site à site avec un état de `available`

```
aws ec2 describe-vpn-connections \
  --filters "Name=state,Values=available"
```

Pour plus d'informations, consultez [Comment fonctionne le VPN de AWS site à site dans le guide de l'utilisateur du VPN](#) de site à site.AWS

- Pour plus de détails sur l'API, reportez-vous [DescribeVpnConnections](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit la connexion VPN spécifiée.

```
Get-EC2VpnConnection -VpnConnectionId vpn-12345678
```

Sortie :

```
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

```

Exemple 2 : Cet exemple décrit toute connexion VPN dont l'état est en attente ou disponible.

```

$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnConnection -Filter $filter

```

Exemple 3 : Cet exemple décrit toutes vos connexions VPN.

```
Get-EC2VpnConnection
```

- Pour plus de détails sur l'API, reportez-vous [DescribeVpnConnections](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DescribeVpnGateways** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DescribeVpnGateways`.

CLI

AWS CLI

Pour décrire vos passerelles privées virtuelles

Cet exemple décrit vos passerelles privées virtuelles.

Commande :

```
aws ec2 describe-vpn-gateways
```

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [DescribeVpnGateways](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple décrit la passerelle privée virtuelle spécifiée.

```
Get-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

Sortie :

```
AvailabilityZone :  
State           : available  
Tags            : {}  
Type            : ipsec.1  
VpcAttachments  : {vpc-12345678}  
VpnGatewayId    : vgw-1a2b3c4d
```

Exemple 2 : Cet exemple décrit toute passerelle privée virtuelle dont l'état est en attente ou disponible.

```
$filter = New-Object Amazon.EC2.Model.Filter  
$filter.Name = "state"  
$filter.Values = @( "pending", "available" )  
  
Get-EC2VpnGateway -Filter $filter
```

Exemple 3 : Cet exemple décrit toutes vos passerelles privées virtuelles.

```
Get-EC2VpnGateway
```

- Pour plus de détails sur l'API, reportez-vous [DescribeVpnGateways](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DetachInternetGateway** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DetachInternetGateway`.

CLI

AWS CLI

Pour détacher une passerelle Internet de votre VPC

L'`detach-internet-gateway` exemple suivant détache la passerelle Internet spécifiée du VPC spécifique.

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

Cette commande ne produit aucun résultat.

Pour plus d'informations, consultez [Passerelles Internet](#) dans le Guide de l'utilisateur Amazon VPC.

- Pour plus de détails sur l'API, reportez-vous [DetachInternetGateway](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple détache la passerelle Internet spécifiée du VPC spécifié.

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

- Pour plus de détails sur l'API, reportez-vous [DetachInternetGateway](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DetachNetworkInterface** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DetachNetworkInterface`.

CLI

AWS CLI

Pour détacher une interface réseau de votre instance

Cet exemple détache l'interface réseau spécifiée de l'instance spécifiée. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 detach-network-interface --attachment-id eni-attach-66c4350a
```

- Pour plus de détails sur l'API, reportez-vous [DetachNetworkInterface](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la pièce jointe spécifiée entre une interface réseau et une instance.

```
Dismount-EC2NetworkInterface -AttachmentId eni-attach-1a2b3c4d -Force
```

- Pour plus de détails sur l'API, reportez-vous [DetachNetworkInterface](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DetachVolume** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DetachVolume`.

CLI

AWS CLI

Pour détacher un volume d'une instance

Cet exemple de commande détache le volume (`vol-049df61146c4d7901`) de l'instance à laquelle il est attaché.

Commande :

```
aws ec2 detach-volume --volume-id vol-1234567890abcdef0
```

Sortie :

```
{
  "AttachTime": "2014-02-27T19:23:06.000Z",
  "InstanceId": "i-1234567890abcdef0",
  "VolumeId": "vol-049df61146c4d7901",
  "State": "detaching",
  "Device": "/dev/sdb"
}
```

- Pour plus de détails sur l'API, reportez-vous [DetachVolume](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : cet exemple détache le volume spécifié.

```
Dismount-EC2Volume -VolumeId vol-12345678
```

Sortie :

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

Exemple 2 : vous pouvez également spécifier l'ID de l'instance et le nom de l'appareil pour vous assurer que vous détachez le volume approprié.

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

- Pour plus de détails sur l'API, reportez-vous [DetachVolume](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DetachVpnGateway** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DetachVpnGateway`.

CLI

AWS CLI

Pour détacher une passerelle privée virtuelle de votre VPC

Cet exemple détache la passerelle privée virtuelle spécifiée du VPC spécifié. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

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

- Pour plus de détails sur l'API, reportez-vous [DetachVpnGateway](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple détache la passerelle privée virtuelle spécifiée du VPC spécifié.

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

- Pour plus de détails sur l'API, reportez-vous [DetachVpnGateway](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DisableVgwRoutePropagation** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DisableVgwRoutePropagation`.

CLI

AWS CLI

Pour désactiver la propagation des itinéraires

Cet exemple empêche la passerelle privée virtuelle spécifiée de propager des itinéraires statiques vers la table de routage spécifiée. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 disable-vgw-route-propagation --route-table-id rtb-22574640 --gateway-id vgw-9a4cacf3
```

- Pour plus de détails sur l'API, reportez-vous [DisableVgwRoutePropagation](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple empêche le VGW de propager automatiquement les itinéraires vers la table de routage spécifiée.

```
Disable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- Pour plus de détails sur l'API, reportez-vous [DisableVgwRoutePropagation](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DisableVpcClassicLink** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DisableVpcClassicLink`.

CLI

AWS CLI

Pour désactiver `ClassicLink` pour un VPC

Cet exemple désactive `ClassicLink` `vpc-88888888`.

Commande :

```
aws ec2 disable-vpc-classic-link --vpc-id vpc-88888888
```

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [DisableVpcClassicLink](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple désactive EC2 `VpcClassicLink` pour le `vpc-01e23c4a5d6db78e9`.

Elle renvoie Vrai ou Faux

```
Disable-EC2VpcClassicLink -VpcId vpc-01e23c4a5d6db78e9
```

- Pour plus de détails sur l'API, reportez-vous [DisableVpcClassicLink](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation `DisableVpcClassicLinkDnsSupport` avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DisableVpcClassicLinkDnsSupport`.

CLI

AWS CLI

Pour désactiver le support ClassicLink DNS pour un VPC

Cet exemple désactive le support ClassicLink DNS pour `vpc-88888888`.

Commande :

```
aws ec2 disable-vpc-classic-link-dns-support --vpc-id vpc-88888888
```

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [DisableVpcClassicLinkDnsSupport](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : cet exemple désactive le support ClassicLink DNS pour le `vpc-0b12d3456a7e8910d`

```
Disable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d
```

- Pour plus de détails sur l'API, reportez-vous [DisableVpcClassicLinkDnsSupport](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DisassociateAddress** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DisassociateAddress`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Démarrer avec les instances](#)

.NET

AWS SDK for .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Disassociate an Elastic IP address from an EC2 instance.
/// </summary>
/// <param name="associationId">The association Id.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DisassociateIp(string associationId)
{
    var response = await _amazonEC2.DisassociateAddressAsync(
        new DisassociateAddressRequest { AssociationId = associationId });
    return response.HttpStatusCode == HttpStatusCode.OK;
}
```

- Pour plus de détails sur l'API, reportez-vous [DisassociateAddress](#) à la section Référence des AWS SDK for .NET API.

Bash

AWS CLI avec le script Bash

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
#####
# 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}" ;;
        esac
    done
}
```

```

        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
}

```

Fonctions utilitaires utilisées dans cet exemple.

```

#####
# 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
}

```

- Pour plus de détails sur l'API, reportez-vous [DisassociateAddress](#) à la section Référence des AWS CLI commandes.

CLI

AWS CLI

Dissociation d'une adresse IP Elastic dans EC2-Classic

Cet exemple dissocie une adresse IP Elastic d'une instance dans EC2-Classic. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 disassociate-address --public-ip 198.51.100.0
```

Dissociation d'une adresse IP Elastic dans EC2-VPC

Cet exemple dissocie une adresse IP Elastic d'une instance dans un VPC. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 disassociate-address --association-id eipassoc-2bebb745
```

- Pour plus de détails sur l'API, reportez-vous [DisassociateAddress](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void disassociateAddress(Ec2Client ec2, String associationId) {
    try {
        DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
            .associationId(associationId)
            .build();

        ec2.disassociateAddress(addressRequest);
        System.out.println("You successfully disassociated the address!");
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DisassociateAddress](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

// Disassociate an Elastic IP address from an instance.
export const main = async () => {
    const command = new DisassociateAddressCommand({
        // You can also use PublicIp, but that is for EC2 classic which is being
        // retired.
        AssociationId: "ASSOCIATION_ID",
    });

    try {
        await client.send(command);
        console.log("Successfully disassociated address");
    } catch (err) {
        console.error(err);
    }
};
```

- Pour plus de détails sur l'API, reportez-vous [DisassociateAddress](#) à la section Référence des AWS SDK for JavaScript API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun disassociateAddressSc(associationIdVal: String?) {
    val addressRequest =
        DisassociateAddressRequest {
            associationId = associationIdVal
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.disassociateAddress(addressRequest)
        println("You successfully disassociated the address!")
    }
}
```

- Pour plus de détails sur l'API, consultez [DisassociateAddress](#) la section AWS SDK pour la référence de l'API Kotlin.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple dissocie l'adresse IP élastique spécifiée de l'instance spécifiée dans un VPC.

```
Unregister-EC2Address -AssociationId eipassoc-12345678
```

Exemple 2 : Cet exemple dissocie l'adresse IP élastique spécifiée de l'instance spécifiée dans EC2-Classic.

```
Unregister-EC2Address -PublicIp 203.0.113.17
```

- Pour plus de détails sur l'API, reportez-vous [DisassociateAddress](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Python

SDK pour Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

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

    def disassociate(self):
        """
        Removes an association between an Elastic IP address and an instance.
        When the
        association is removed, the instance is assigned a new public IP address.
```

```
"""
    if self.elastic_ip is None:
        logger.info("No Elastic IP to disassociate.")
        return

    try:
        self.elastic_ip.association.delete()
    except ClientError as err:
        logger.error(
            "Couldn't disassociate Elastic IP %s from its instance. Here's
why: %s: %s",
            self.elastic_ip.allocation_id,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

- Pour plus de détails sur l'API, consultez [DisassociateAddress](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **DisassociateRouteTable** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `DisassociateRouteTable`.

CLI

AWS CLI

Pour dissocier une table de routage

Cet exemple dissocie la table de routage spécifiée du sous-réseau spécifié. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 disassociate-route-table --association-id rtbassoc-781d0d1a
```

- Pour plus de détails sur l'API, reportez-vous [DisassociateRouteTable](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple supprime l'association spécifiée entre une table de routage et un sous-réseau.

```
Unregister-EC2RouteTable -AssociationId rtbassoc-1a2b3c4d
```

- Pour plus de détails sur l'API, reportez-vous [DisassociateRouteTable](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **EnableVgwRoutePropagation** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `EnableVgwRoutePropagation`.

CLI

AWS CLI

Pour activer la propagation des itinéraires

Cet exemple permet à la passerelle privée virtuelle spécifiée de propager des itinéraires statiques vers la table de routage spécifiée. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 enable-vgw-route-propagation --route-table-id rtb-22574640 --gateway-id vgw-9a4cacf3
```

- Pour plus de détails sur l'API, reportez-vous [EnableVgwRoutePropagation](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple permet au VGW spécifié de propager automatiquement les itinéraires vers la table de routage spécifiée.

```
Enable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- Pour plus de détails sur l'API, reportez-vous [EnableVgwRoutePropagation](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **EnableVolumeIo** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `EnableVolumeIo`.

CLI

AWS CLI

Pour activer les E/S pour un volume

Cet exemple active les E/S sur le volume `vol-1234567890abcdef0`.

Commande :

```
aws ec2 enable-volume-io --volume-id vol-1234567890abcdef0
```

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [EnableVolumeIo](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple active les opérations d'E/S pour le volume spécifié, si les opérations d'E/S ont été désactivées.

```
Enable-EC2VolumeIO -VolumeId vol-12345678
```

- Pour plus de détails sur l'API, reportez-vous [EnableVolumeIO](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **EnableVpcClassicLink** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `EnableVpcClassicLink`.

CLI

AWS CLI

Pour activer un VPC pour ClassicLink

Cet exemple active `vpc-88888888` pour ClassicLink

Commande :

```
aws ec2 enable-vpc-classic-link --vpc-id vpc-88888888
```

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [EnableVpcClassicLink](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple active le VPC `vpc-0123456b789b0d12f` pour ClassicLink

```
Enable-EC2VpcClassicLink -VpcId vpc-0123456b789b0d12f
```

Sortie :

```
True
```

- Pour plus de détails sur l'API, reportez-vous [EnableVpcClassicLink](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **EnableVpcClassicLinkDnsSupport** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `EnableVpcClassicLinkDnsSupport`.

CLI

AWS CLI

Pour activer le support ClassicLink DNS pour un VPC

Cet exemple active le support ClassicLink DNS pour `vpc-88888888`.

Commande :

```
aws ec2 enable-vpc-classic-link-dns-support --vpc-id vpc-88888888
```

Sortie :

```
{
```

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

- Pour plus de détails sur l'API, reportez-vous [EnableVpcClassicLinkDnsSupport](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple permet à vpc-0b12d3456a7e8910d de prendre en charge la résolution des noms d'hôte DNS pour ClassicLink

```
Enable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

- Pour plus de détails sur l'API, reportez-vous [EnableVpcClassicLinkDnsSupport](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **GetConsoleOutput** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `GetConsoleOutput`.

CLI

AWS CLI

Exemple 1 : pour obtenir la sortie de la console

L'`get-console-output` exemple suivant obtient la sortie de console pour l'instance Linux spécifiée.

```
aws ec2 get-console-output \  
  --instance-id i-1234567890abcdef0
```

Sortie :

```
{
  "InstanceId": "i-1234567890abcdef0",
  "Timestamp": "2013-07-25T21:23:53.000Z",
  "Output": "...
}
```

Pour plus d'informations, consultez [la sortie de la console Instance](#) dans le guide de l'utilisateur Amazon EC2.

Exemple 2 : pour obtenir la dernière sortie de console

L'`get-console-output`exemple suivant obtient la dernière sortie de console pour l'instance Linux spécifiée.

```
aws ec2 get-console-output \
  --instance-id i-1234567890abcdef0 \
  --latest \
  --output text
```

Sortie :

```
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
```

Pour plus d'informations, consultez [la sortie de la console Instance](#) dans le guide de l'utilisateur Amazon EC2.

- Pour plus de détails sur l'API, reportez-vous [GetConsoleOutput](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple obtient la sortie de console pour l'instance Linux spécifiée. La sortie de la console est codée.

```
Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456
```

Sortie :

InstanceId	Output
-----	-----
i-0e194d3c47c123637	WyAgICAwLjAwMDAwMF0gQ29tbW...bGU9dHR5UzAgc2Vs

Exemple 2 : Cet exemple stocke la sortie codée de la console dans une variable, puis la décode.

```
$Output_encoded = (Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456).Output  
[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($Output_encoded))
```

- Pour plus de détails sur l'API, reportez-vous [GetConsoleOutput](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **GetHostReservationPurchasePreview** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `GetHostReservationPurchasePreview`.

CLI

AWS CLI

Pour obtenir un aperçu de l'achat d'une réservation d'hôte dédié

Cet exemple fournit un aperçu des coûts d'une réservation d'hôte dédié spécifiée pour l'hôte dédié spécifié dans votre compte.

Commande :

```
aws ec2 get-host-reservation-purchase-preview --offering-id hro-03f707bf363b6b324
--host-id-set h-013abcd2a00cbd123
```

Sortie :

```
{
  "TotalHourlyPrice": "1.499",
  "Purchase": [
    {
      "HourlyPrice": "1.499",
      "InstanceFamily": "m4",
      "PaymentOption": "NoUpfront",
      "HostIdSet": [
        "h-013abcd2a00cbd123"
      ],
      "UpfrontPrice": "0.000",
      "Duration": 31536000
    }
  ],
  "TotalUpfrontPrice": "0.000"
}
```

- Pour plus de détails sur l'API, reportez-vous [GetHostReservationPurchasePreview](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple affiche un aperçu d'un achat de réservation avec des configurations qui correspondent à celles de votre hôte dédié h-01e23f4cd567890f1

```
Get-EC2HostReservationPurchasePreview -OfferingId hro-0c1f23456789d0ab -HostIdSet
h-01e23f4cd567890f1
```

Sortie :

```

CurrencyCode Purchase TotalHourlyPrice TotalUpfrontPrice
-----
{}          1.307          0.000

```

- Pour plus de détails sur l'API, reportez-vous [GetHostReservationPurchasePreview](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **GetPasswordData** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `GetPasswordData`.

CLI

AWS CLI

Pour obtenir le mot de passe crypté

Cet exemple permet d'obtenir le mot de passe crypté.

Commande :

```
aws ec2 get-password-data --instance-id i-1234567890abcdef0
```

Sortie :

```
{
  "InstanceId": "i-1234567890abcdef0",
  "Timestamp": "2013-08-07T22:18:38.000Z",
  "PasswordData": "gSlJFq+VpcZXqy+iktxMF6NyxQ4qCrT4+ga0uN0enX1MmgXPTj7XEXAMPLE
UQ+YeFfb+L1U4C4AKv652Ux1iRB3CPTY7WmU3TUnhsuBd+p6LVk7T21KUm160Xbk6WPW1VYYm/TRPB1
e1DQ7PY4an/DgZT4mwcpRFigzhniQgDDe01InvSDcwoUTwNs0Y1S8ouri2W4n5GNlriM3Q0AnNVe1Vz/
53TkDtxbNoU606M1gK9zUWSxqEgwbvV2j8c5rP0WCuaMWSF14ziDu4bd7q+4RSyi8NUsVWnKZ4aEZffu
DPGzKrF5yLlf3etP2L4ZR6CvG7K1hx7VK0QVN32Dajw=="
}
```

Pour obtenir le mot de passe déchiffré

Cet exemple permet d'obtenir le mot de passe déchiffré.

Commande :

```
aws ec2 get-password-data --instance-id i-1234567890abcdef0 --priv-launch-key C:\Keys\MyKeyPair.pem
```

Sortie :

```
{
  "InstanceId": "i-1234567890abcdef0",
  "Timestamp": "2013-08-30T23:18:05.000Z",
  "PasswordData": "&ViJ652e*u"
}
```

- Pour plus de détails sur l'API, reportez-vous [GetPasswordData](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple déchiffre le mot de passe attribué par Amazon EC2 au compte administrateur pour l'instance Windows spécifiée. Lorsqu'un fichier pem a été spécifié, le paramètre du commutateur `-Decrypt` est automatiquement supposé.

```
Get-EC2PasswordData -InstanceId i-12345678 -PemFile C:\path\my-key-pair.pem
```

Sortie :

```
mYZ(PA9?C)Q
```

Exemple 2 : (Windows PowerShell uniquement) Inspecte l'instance pour déterminer le nom de la paire de clés utilisée pour lancer l'instance, puis tente de trouver les données de paire de clés correspondantes dans le magasin de configuration du Toolkit for Visual Studio AWS . Si les données de la paire de clés sont trouvées, le mot de passe est déchiffré.

```
Get-EC2PasswordData -InstanceId i-12345678 -Decrypt
```

Sortie :

```
mYZ(PA9?C)Q
```

Exemple 3 : renvoie les données de mot de passe chiffrées pour l'instance.

```
Get-EC2PasswordData -InstanceId i-12345678
```

Sortie :

```
iVz3BAK/WAXV.....dqt8WeMA==
```

- Pour plus de détails sur l'API, reportez-vous [GetPasswordData](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **ImportImage** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ImportImage`.

CLI

AWS CLI

Pour importer un fichier image de machine virtuelle en tant qu'AMI

L'`import-image` exemple suivant importe le fichier OVA spécifié.

```
aws ec2 import-image \  
  --disk-containers Format=ova,UserBucket="{S3Bucket=my-import-bucket,S3Key=vms/  
my-server-vm.ova}"
```

Sortie :

```
{  
  "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"
}

```

- Pour plus de détails sur l'API, reportez-vous [ImportImage](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple importe une image de machine virtuelle à disque unique depuis le compartiment Amazon S3 spécifié vers Amazon EC2 avec un jeton d'idempotence. L'exemple nécessite qu'un rôle de service d'importation de machine virtuelle portant le nom par défaut « vmimport » existe, avec une politique autorisant Amazon EC2 à accéder au compartiment spécifié, comme expliqué dans la rubrique Prérequis pour l'importation de machines virtuelles. Pour utiliser un rôle personnalisé, spécifiez le nom du rôle à l'aide du **-RoleName** paramètre.

```

$container = New-Object Amazon.EC2.Model.ImageDiskContainer
$container.Format="VMDK"
$container.UserBucket = New-Object Amazon.EC2.Model.UserBucket
$container.UserBucket.S3Bucket = "myVirtualMachineImages"
$container.UserBucket.S3Key = "Win_2008_Server_Standard_SP2_64-bit-disk1.vmdk"

$params = @{
    "ClientToken"="idempotencyToken"
    "Description"="Windows 2008 Standard Image Import"
    "Platform"="Windows"
    "LicenseType"="AWS"
}

Import-EC2Image -DiskContainer $container @params

```

Sortie :

```
Architecture      :  
Description       : Windows 2008 Standard Image  
Hypervisor        :  
ImageId           :  
ImportTaskId      : import-ami-abcdefgh  
LicenseType       : AWS  
Platform          : Windows  
Progress          : 2  
SnapshotDetails   : {}  
Status            : active  
StatusMessage     : pending
```

- Pour plus de détails sur l'API, reportez-vous [ImportImage](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **ImportKeyPair** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ImportKeyPair`.

CLI

AWS CLI

Pour importer une clé publique

Commencez par générer une paire de clés avec l'outil de votre choix. Par exemple, utilisez cette commande `ssh-keygen` :

Commande :

```
ssh-keygen -t rsa -C "my-key" -f ~/.ssh/my-key
```

Sortie :

```
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.
...
```

Cet exemple de commande importe la clé publique spécifiée.

Commande :

```
aws ec2 import-key-pair --key-name "my-key" --public-key-material file://~/.ssh/
my-key.pub
```

Sortie :

```
{
  "KeyName": "my-key",
  "KeyFingerprint": "1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca"
}
```

- Pour plus de détails sur l'API, reportez-vous [ImportKeyPair](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple importe une clé publique dans EC2. La première ligne stocke le contenu du fichier de clé publique (*.pub) dans la variable **\$publickey**. Ensuite, l'exemple convertit le format UTF8 du fichier de clé publique en une chaîne codée en Base64 et stocke la chaîne convertie dans la variable. **\$pkbase64** Dans la dernière ligne, la clé publique convertie est importée dans EC2. L'applet de commande renvoie l'empreinte digitale et le nom de la clé en tant que résultats.

```
$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
```

Sortie :

KeyFingerprint	KeyName
-----	-----
do:d0:15:8f:79:97:12:be:00:fd:df:31:z3:b1:42:z1	Example-user-key

- Pour plus de détails sur l'API, reportez-vous [ImportKeyPair](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **ImportSnapshot** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ImportSnapshot`.

CLI

AWS CLI

Pour importer un instantané

L'`import-snapshot` exemple suivant importe le disque spécifié sous forme de capture instantanée.

```
aws ec2 import-snapshot \
  --description "My server VMDK" \
  --disk-container Format=VMDK,UserBucket={S3Bucket=my-import-bucket,S3Key=vms/
my-server-vm.vmdk}
```

Sortie :

```
{
  "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"
    }
  }
}

```

- Pour plus de détails sur l'API, reportez-vous [ImportSnapshot](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple importe une image de disque de machine virtuelle au format « VMDK » dans un instantané Amazon EBS. L'exemple nécessite un rôle de service d'importation de machine virtuelle portant le nom par défaut « vmimport », avec une politique autorisant Amazon EC2 à accéder au bucket spécifié, comme expliqué dans **VM Import Prerequisites** la rubrique AWSEC <http://docs.aws.amazon.com/WindowsGuide/ImportPrerequisites/2/latest/VM.html>. Pour utiliser un rôle personnalisé, spécifiez le nom du rôle à l'aide du **-RoleName** paramètre.

```

$params = @{
  "ClientToken"="idempotencyToken"
  "Description"="Disk Image Import"
  "DiskContainer_Description" = "Data disk"
  "DiskContainer_Format" = "VMDK"
  "DiskContainer_S3Bucket" = "myVirtualMachineImages"
  "DiskContainer_S3Key" = "datadiskimage.vmdk"
}

Import-EC2Snapshot @params

```

Sortie :

Description	ImportTaskId	SnapshotTaskDetail
-----	-----	-----

```
Disk Image Import      import-snap-abcdefgh  
Amazon.EC2.Model.SnapshotTaskDetail
```

- Pour plus de détails sur l'API, reportez-vous [ImportSnapshot](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **ModifyCapacityReservation** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ModifyCapacityReservation`.

CLI

AWS CLI

Exemple 1 : pour modifier le nombre d'instances réservées par une réservation de capacité existante

L'`modify-capacity-reservation` exemple suivant modifie le nombre d'instances pour lesquelles la réservation de capacité réserve de la capacité.

```
aws ec2 modify-capacity-reservation \  
  --capacity-reservation-id cr-1234abcd56EXAMPLE \  
  --instance-count 5
```

Sortie :

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

Exemple 2 : pour modifier la date et l'heure de fin d'une réservation de capacité existante

L'`modify-capacity-reservation` exemple suivant modifie une réservation de capacité existante pour qu'elle se termine à la date et à l'heure spécifiées.

```
aws ec2 modify-capacity-reservation \  
  --capacity-reservation-id cr-1234abcd56EXAMPLE \  
  --end-time 2017-01-01T00:00:00Z
```

```
--capacity-reservation-id cr-1234abcd56EXAMPLE \  
--end-date-type limited \  
--end-date 2019-08-31T23:59:59Z
```

Pour plus d'informations, consultez [Modifier une réservation de capacité](#) dans le guide de l'utilisateur Amazon Elastic Compute Cloud pour les instances Linux.

- Pour plus de détails sur l'API, reportez-vous [ModifyCapacityReservation](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple modifie le CapacityReservationId cr-0c1f2345db6f7cdba en remplaçant le nombre d'instances par 1

```
Edit-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba -  
InstanceCount 1
```

Sortie :

```
True
```

- Pour plus de détails sur l'API, reportez-vous [ModifyCapacityReservation](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **ModifyHosts** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ModifyHosts`.

CLI

AWS CLI

Exemple 1 : pour activer le placement automatique pour un hôte dédié

L'`modify-hostsexemple` suivant active le placement automatique d'un hôte dédié afin qu'il accepte tout lancement d'instance non ciblé correspondant à la configuration de son type d'instance.

```
aws ec2 modify-hosts \  
  --host-id h-06c2f189b4EXAMPLE \  
  --auto-placement on
```

Sortie :

```
{  
  "Successful": [  
    "h-06c2f189b4EXAMPLE"  
  ],  
  "Unsuccessful": []  
}
```

Exemple 2 : Pour activer la restauration d'un hôte dédié

L'`modify-hostsexemple` suivant active la restauration de l'hôte pour l'hôte dédié spécifié.

```
aws ec2 modify-hosts \  
  --host-id h-06c2f189b4EXAMPLE \  
  --host-recovery on
```

Sortie :

```
{  
  "Successful": [  
    "h-06c2f189b4EXAMPLE"  
  ],  
  "Unsuccessful": []  
}
```

Pour plus d'informations, consultez [Modifier le placement automatique des hôtes dédiés](#) dans le guide de l'utilisateur Amazon Elastic Compute Cloud pour les instances Linux.

- Pour plus de détails sur l'API, reportez-vous [ModifyHosts](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : cet exemple modifie les AutoPlacement paramètres sur off pour l'hôte dédié h-01e23f4cd567890f3

```
Edit-EC2Host -HostId h-03e09f8cd681609f3 -AutoPlacement off
```

Sortie :

```
Successful          Unsuccessful
-----
{h-01e23f4cd567890f3} {}
```

- Pour plus de détails sur l'API, reportez-vous [ModifyHosts](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **ModifyIdFormat** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ModifyIdFormat`.

CLI

AWS CLI

Pour activer le format d'identifiant plus long pour une ressource

L'`modify-id-format` exemple suivant active le format d'identifiant plus long pour le type de `instance` ressource.

```
aws ec2 modify-id-format \  
  --resource instance \  
  --use-long-ids
```

Pour désactiver le format d'identifiant plus long pour une ressource

L'`modify-id-format` exemple suivant désactive le format d'identifiant long pour le type de instance ressource.

```
aws ec2 modify-id-format \  
  --resource instance \  
  --no-use-long-ids
```

L'`modify-id-format` exemple suivant active le format d'identifiant plus long pour tous les types de ressources pris en charge dont la période d'inscription est en cours.

```
aws ec2 modify-id-format \  
  --resource all-current \  
  --use-long-ids
```

- Pour plus de détails sur l'API, reportez-vous [ModifyIdFormat](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple active le format d'identifiant plus long pour le type de ressource spécifié.

```
Edit-EC2IdFormat -Resource instance -UseLongId $true
```

Exemple 2 : cet exemple désactive le format d'identifiant plus long pour le type de ressource spécifié.

```
Edit-EC2IdFormat -Resource instance -UseLongId $false
```

- Pour plus de détails sur l'API, reportez-vous [ModifyIdFormat](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation `ModifyImageAttribute` avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ModifyImageAttribute`.

CLI

AWS CLI

Exemple 1 : Pour rendre publique une AMI

L'`modify-instance-attribute` exemple suivant rend publique l'AMI spécifiée.

```
aws ec2 modify-image-attribute \  
  --image-id ami-5731123e \  
  --launch-permission "Add=[{Group=all}]"
```

Cette commande ne produit aucun résultat.

Exemple 2 : pour rendre une AMI privée

L'`modify-instance-attribute` exemple suivant rend l'AMI spécifiée privée.

```
aws ec2 modify-image-attribute \  
  --image-id ami-5731123e \  
  --launch-permission "Remove=[{Group=all}]"
```

Cette commande ne produit aucun résultat.

Exemple 3 : Pour accorder une autorisation de lancement à un AWS compte

L'`modify-instance-attribute` exemple suivant accorde des autorisations de lancement au AWS compte spécifié.

```
aws ec2 modify-image-attribute \  
  --image-id ami-5731123e \  
  --launch-permission "Add=[{UserId=123456789012}]"
```

Cette commande ne produit aucun résultat.

Exemple 4 : pour supprimer l'autorisation de lancement d'un AWS compte

L'`modify-instance-attribute` exemple suivant supprime les autorisations de lancement du AWS compte spécifié.

```
aws ec2 modify-image-attribute \  
  --image-id ami-5731123e \  
  --launch-permission "Remove=[{UserId=123456789012}]"
```

- Pour plus de détails sur l'API, reportez-vous [ModifyImageAttribute](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple met à jour la description de l'AMI spécifiée.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Description "New description"
```

Exemple 2 : Cet exemple rend l'AMI publique (par exemple, afin que tout le Compte AWS monde puisse l'utiliser).

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType add -UserGroup all
```

Exemple 3 : Cet exemple rend l'AMI privée (par exemple, afin que vous soyez le seul à pouvoir l'utiliser en tant que propriétaire).

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType remove -UserGroup all
```

Exemple 4 : Cet exemple accorde l'autorisation de lancement à la personne spécifiée Compte AWS.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType add -UserId 111122223333
```

Exemple 5 : Cet exemple supprime l'autorisation de lancement spécifiée Compte AWS.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType remove -UserId 111122223333
```

- Pour plus de détails sur l'API, reportez-vous [ModifyImageAttribute](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **ModifyInstanceAttribute** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ModifyInstanceAttribute`.

CLI

AWS CLI

Exemple 1 : pour modifier le type d'instance

L'`modify-instance-attribute` exemple suivant modifie le type d'instance de l'instance spécifiée. L'instance doit être dans l'état `stopped`.

```
aws ec2 modify-instance-attribute \  
  --instance-id i-1234567890abcdef0 \  
  --instance-type "{\"Value\": \"m1.small\"}"
```

Cette commande ne produit aucun résultat.

Exemple 2 : pour activer la mise en réseau améliorée sur une instance

L'`modify-instance-attribute` exemple suivant permet d'améliorer la mise en réseau pour l'instance spécifiée. L'instance doit être dans l'état `stopped`.

```
aws ec2 modify-instance-attribute \  
  --instance-id i-1234567890abcdef0 \  
  --sriov-net-support simple
```

Cette commande ne produit aucun résultat.

Exemple 3 : pour modifier l' `sourceDestCheck` attribut

L'`modify-instance-attribute` exemple suivant définit l'`sourceDestCheck` attribut de l'instance spécifiée sur `true`. L'instance doit se trouver dans un VPC.

```
aws ec2 modify-instance-attribute --instance-id i-1234567890abcdef0 --source-dest-check "{\"Value\": true}"
```

Cette commande ne produit aucun résultat.

Exemple 4 : pour modifier l' `deleteOnTermination` attribut du volume racine

L'`modify-instance-attribute` exemple suivant définit l'`deleteOnTermination` attribut du volume racine de l'instance basée sur Amazon EBS spécifiée sur `false`. Par défaut, cet attribut est `true` destiné au volume racine.

Commande :

```
aws ec2 modify-instance-attribute \
  --instance-id i-1234567890abcdef0 \
  --block-device-mappings "[{\"DeviceName\": \"/dev/sda1\", \"Ebs\": {\"DeleteOnTermination\": false}}]"
```

Cette commande ne produit aucun résultat.

Exemple 5 : pour modifier les données utilisateur associées à une instance

L'`modify-instance-attribute` exemple suivant ajoute le contenu du fichier `UserData.txt` tel que celui `UserData` de l'instance spécifiée.

Contenu du fichier original `UserData.txt` :

```
#!/bin/bash
yum update -y
service httpd start
chkconfig httpd on
```

Le contenu du fichier doit être codé en base64. La première commande convertit le fichier texte en base64 et l'enregistre en tant que nouveau fichier.

Version Linux/macOS de la commande :

```
base64 UserData.txt > UserData.base64.txt
```

Cette commande ne produit aucun résultat.

Version Windows de la commande :

```
certutil -encode UserData.txt tmp.b64 && findstr /v /c:- tmp.b64 >
UserData.base64.txt
```

Sortie :

```
Input Length = 67
Output Length = 152
CertUtil: -encode command completed successfully.
```

Vous pouvez maintenant référencer ce fichier dans la commande CLI suivante :

```
aws ec2 modify-instance-attribute \
  --instance-id=i-09b5a14dbca622e76 \
  --attribute userData --value file://UserData.base64.txt
```

Cette commande ne produit aucun résultat.

Pour plus d'informations, consultez les [données utilisateur et la AWS CLI](#) dans le guide de l'utilisateur EC2.

- Pour plus de détails sur l'API, reportez-vous [ModifyInstanceAttribute](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple modifie le type d'instance de l'instance spécifiée.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceType m3.medium
```

Exemple 2 : Cet exemple permet d'améliorer la mise en réseau pour l'instance spécifiée, en spécifiant « simple » comme valeur du paramètre de support réseau de virtualisation des E/S à racine unique (SR-IOV), -. SriovNetSupport

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SriovNetSupport "simple"
```

Exemple 3 : Cet exemple modifie les groupes de sécurité pour l'instance spécifiée. L'instance doit se trouver dans un VPC. Vous devez spécifier l'ID de chaque groupe de sécurité, et non le nom.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -Group @( "sg-12345678",  
"sg-45678901" )
```

Exemple 4 : Cet exemple active l'optimisation des E/S EBS pour l'instance spécifiée. Cette fonctionnalité n'est pas disponible pour tous les types d'instances. Des frais d'utilisation supplémentaires s'appliquent lors de l'utilisation d'une instance optimisée pour EBS.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -EbsOptimized $true
```

Exemple 5 : Cet exemple active la vérification de la source/de la destination pour l'instance spécifiée. Pour qu'une instance NAT exécute le NAT, la valeur doit être « false ».

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SourceDestCheck $true
```

Exemple 6 : Cet exemple désactive la résiliation pour l'instance spécifiée.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -DisableApiTermination $true
```

Exemple 7 : Cet exemple modifie l'instance spécifiée afin qu'elle s'arrête lorsque l'arrêt est lancé à partir de l'instance.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -  
InstanceInitiatedShutdownBehavior terminate
```

- Pour plus de détails sur l'API, reportez-vous [ModifyInstanceAttribute](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **ModifyInstanceCreditSpecification** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ModifyInstanceCreditSpecification`.

CLI

AWS CLI

Pour modifier l'option de crédit pour l'utilisation du processeur d'une instance

Cet exemple modifie l'option de crédit pour l'utilisation du processeur de l'instance spécifiée dans la région spécifiée en « illimitée ». Les options de crédit valides sont « standard » et « illimitées ».

Commande :

```
aws ec2 modify-instance-credit-specification --instance-credit-specification
"InstanceId=i-1234567890abcdef,CpuCredits=unlimited"
```

Sortie :

```
{
  "SuccessfulInstanceCreditSpecifications": [
    {
      "InstanceId": "i-1234567890abcdef"
    }
  ],
  "UnsuccessfulInstanceCreditSpecifications": []
}
```

- Pour plus de détails sur l'API, reportez-vous [ModifyInstanceCreditSpecification](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cela active les crédits T2 illimités, par exemple i-01234567890abcdef.

```
$Credit = New-Object -TypeName
Amazon.EC2.Model.InstanceCreditSpecificationRequest
$Credit.InstanceId = "i-01234567890abcdef"
$Credit.CpuCredits = "unlimited"
Edit-EC2InstanceCreditSpecification -InstanceCreditSpecification $Credit
```

- Pour plus de détails sur l'API, reportez-vous [ModifyInstanceCreditSpecification](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **ModifyNetworkInterfaceAttribute** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ModifyNetworkInterfaceAttribute`.

CLI

AWS CLI

Pour modifier l'attribut de pièce jointe d'une interface réseau

Cet exemple de commande modifie l'`attachment` attribut de l'interface réseau spécifiée.

Commande :

```
aws ec2 modify-network-interface-attribute --network-interface-id eni-686ea200 --attachment AttachmentId=eni-attach-43348162,DeleteOnTermination=false
```

Pour modifier l'attribut de description d'une interface réseau

Cet exemple de commande modifie l'`description` attribut de l'interface réseau spécifiée.

Commande :

```
aws ec2 modify-network-interface-attribute --network-interface-id eni-686ea200 --description "My description"
```

Pour modifier l'attribut `GroupSet` d'une interface réseau

Cet exemple de commande modifie l'`groupSet` attribut de l'interface réseau spécifiée.

Commande :

```
aws ec2 modify-network-interface-attribute --network-interface-id eni-686ea200 --groups sg-903004f8 sg-1a2b3c4d
```

Pour modifier l' attribut sourceDestCheck d'une interface réseau

Cet exemple de commande modifie l'attribut sourceDestCheck de l'interface réseau spécifiée.

Commande :

```
aws ec2 modify-network-interface-attribute --network-interface-id eni-686ea200 --no-source-dest-check
```

- Pour plus de détails sur l'API, reportez-vous [ModifyNetworkInterfaceAttribute](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple modifie l'interface réseau spécifiée afin que la pièce jointe spécifiée soit supprimée lors de la résiliation.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Attachment_AttachmentId eni-attach-1a2b3c4d -Attachment_DeleteOnTermination $true
```

Exemple 2 : Cet exemple modifie la description de l'interface réseau spécifiée.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Description "my description"
```

Exemple 3 : Cet exemple modifie le groupe de sécurité pour l'interface réseau spécifiée.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Groups sg-1a2b3c4d
```

Exemple 4 : Cet exemple désactive la vérification source/destination pour l'interface réseau spécifiée.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -  
SourceDestCheck $false
```

- Pour plus de détails sur l'API, reportez-vous [ModifyNetworkInterfaceAttribute](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **ModifyReservedInstances** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ModifyReservedInstances`.

CLI

AWS CLI

Pour modifier les instances réservées

Cet exemple de commande déplace une instance réservée vers une autre zone de disponibilité de la même région.

Commande :

```
aws ec2 modify-reserved-instances --reserved-instances-ids b847fa93-  
e282-4f55-b59a-1342f5bd7c02 --target-configurations AvailabilityZone=us-  
west-1c,Platform=EC2-Classic,InstanceCount=10
```

Sortie :

```
{  
  "ReservedInstancesModificationId": "rimod-d3ed4335-b1d3-4de6-ab31-0f13aaf46687"  
}
```

Pour modifier la plate-forme réseau des instances réservées

Cet exemple de commande convertit les instances réservées EC2-Classic en EC2-VPC.

Commande :

```
aws ec2 modify-reserved-instances --reserved-instances-ids f127bd27-  
edb7-44c9-a0eb-0d7e09259af0 --target-configurations AvailabilityZone=us-  
west-1c,Platform=EC2-VPC,InstanceCount=5
```

Sortie :

```
{  
  "ReservedInstancesModificationId": "rimod-82fa9020-668f-4fb6-945d-61537009d291"  
}
```

Pour plus d'informations, consultez la section Modification de vos instances réservées dans le guide de l'utilisateur Amazon EC2.

Pour modifier la taille d'instance des instances réservées

Cet exemple de commande modifie une instance réservée qui possède 10 instances Linux/UNIX m1.small dans us-west-1c afin que 8 instances m1.small deviennent 2 instances m1.large, et les 2 m1.small restantes deviennent 1 instance m1.medium dans la même zone de disponibilité. Commande :

```
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
```

Sortie :

```
{  
  "ReservedInstancesModificationId": "rimod-acc5f240-080d-4717-  
b3e3-1c6b11fa00b6"  
}
```

Pour plus d'informations, consultez la section Modification de la taille de l'instance de vos réservations dans le guide de l'utilisateur Amazon EC2.

- Pour plus de détails sur l'API, reportez-vous [ModifyReservedInstances](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple modifie la zone de disponibilité, le nombre d'instances et la plateforme pour les instances réservées spécifiées.

```
$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
```

- Pour plus de détails sur l'API, reportez-vous [ModifyReservedInstances](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **ModifySnapshotAttribute** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ModifySnapshotAttribute`.

CLI

AWS CLI

Exemple 1 : pour modifier un attribut de capture d'écran

L'`modify-snapshot-attribute` exemple suivant met à jour l'`createVolumePermission` attribut pour le cliché spécifié, en supprimant les autorisations de volume pour l'utilisateur spécifié.

```
aws ec2 modify-snapshot-attribute \
  --snapshot-id snap-1234567890abcdef0 \
  --attribute createVolumePermission \
  --operation-type remove \
```

```
--user-ids 123456789012
```

Exemple 2 : Pour rendre public un instantané

L'`modify-snapshot-attribute` exemple suivant rend public le cliché spécifié.

```
aws ec2 modify-snapshot-attribute \  
  --snapshot-id snap-1234567890abcdef0 \  
  --attribute createVolumePermission \  
  --operation-type add \  
  --group-names all
```

- Pour plus de détails sur l'API, reportez-vous [ModifySnapshotAttribute](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple rend public le cliché spécifié en définissant son `CreateVolumePermission` attribut.

```
Edit-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute  
CreateVolumePermission -OperationType Add -GroupName all
```

- Pour plus de détails sur l'API, reportez-vous [ModifySnapshotAttribute](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **ModifySpotFleetRequest** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ModifySpotFleetRequest`.

CLI

AWS CLI

Pour modifier une demande de flotte Spot

Cet exemple de commande met à jour la capacité cible de la demande de parc Spot spécifiée.

Commande :

```
aws ec2 modify-spot-fleet-request --target-capacity 20 --spot-fleet-request-id
sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
```

Sortie :

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

Cet exemple de commande réduit la capacité cible de la demande de parc Spot spécifiée sans mettre fin à aucune instance Spot en conséquence.

Commande :

```
aws ec2 modify-spot-fleet-request --target-capacity 10 --excess-capacity-
termination-policy NoTermination --spot-fleet-request-ids sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE
```

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [ModifySpotFleetRequest](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple met à jour la capacité cible de la demande de parc Spot spécifiée.

```
Edit-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TargetCapacity 10
```

Sortie :

```
True
```

- Pour plus de détails sur l'API, reportez-vous [ModifySpotFleetRequest](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **ModifySubnetAttribute** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ModifySubnetAttribute`.

CLI

AWS CLI

Pour modifier le comportement d'adressage IPv4 public d'un sous-réseau

Cet exemple modifie le sous-réseau -1a2b3c4d pour spécifier qu'une adresse IPv4 publique est attribuée à toutes les instances lancées dans ce sous-réseau. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 modify-subnet-attribute --subnet-id subnet-1a2b3c4d --map-public-ip-on-launch
```

Pour modifier le comportement d'adressage IPv6 d'un sous-réseau

Cet exemple modifie le sous-réseau -1a2b3c4d pour spécifier que toutes les instances lancées dans ce sous-réseau se voient attribuer une adresse IPv6 provenant de la plage du sous-réseau.

Commande :

```
aws ec2 modify-subnet-attribute --subnet-id subnet-1a2b3c4d --assign-ipv6-address-on-creation
```

Pour plus d'informations, consultez la section Adressage IP dans votre VPC dans le Guide de l'utilisateur du cloud privé AWS virtuel.

- Pour plus de détails sur l'API, reportez-vous [ModifySubnetAttribute](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple active l'adressage IP public pour le sous-réseau spécifié.

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $true
```

Exemple 2 : Cet exemple désactive l'adressage IP public pour le sous-réseau spécifié.

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $false
```

- Pour plus de détails sur l'API, reportez-vous [ModifySubnetAttribute](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **ModifyVolumeAttribute** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ModifyVolumeAttribute`.

CLI

AWS CLI

Pour modifier un attribut de volume

Cet exemple définit l'`autoEnableIo` attribut du volume avec l'ID `vol-1234567890abcdef0` sur `true`. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 modify-volume-attribute --volume-id vol-1234567890abcdef0 --auto-enable-  
io
```

- Pour plus de détails sur l'API, reportez-vous [ModifyVolumeAttribute](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple modifie l'attribut spécifié du volume spécifié. Les opérations d'E/S pour le volume reprennent automatiquement après avoir été suspendues en raison de données potentiellement incohérentes.

```
Edit-EC2VolumeAttribute -VolumeId vol-12345678 -AutoEnableIO $true
```

- Pour plus de détails sur l'API, reportez-vous [ModifyVolumeAttribute](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **ModifyVpcAttribute** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ModifyVpcAttribute`.

CLI

AWS CLI

Pour modifier l' `enableDnsSupport` attribut

Cet exemple modifie l'`enableDnsSupport` attribut. Cet attribut indique si la résolution DNS est activée pour le VPC. Si cet attribut est `true`, le serveur Amazon DNS résout les noms d'hôte DNS de vos instances par rapport aux adresses IP correspondantes. Dans le cas contraire, il ne le fait pas. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 modify-vpc-attribute --vpc-id vpc-a01106c2 --enable-dns-support "{\"Value\":false}"
```

Pour modifier l' `enableDnsHostnames` attribut

Cet exemple modifie l'`enableDnsHostnames` attribut. Cet attribut indique si les instances lancées dans le VPC obtiennent des noms d'hôte DNS. Si l'attribut est défini sur `true`, les instances incluses dans le VPC ont obtenu des noms d'hôte DNS. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 modify-vpc-attribute --vpc-id vpc-a01106c2 --enable-dns-hostnames "{\"Value\":false}"
```

- Pour plus de détails sur l'API, reportez-vous [ModifierVpcAttribute](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple active la prise en charge des noms d'hôte DNS pour le VPC spécifié.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $true
```

Exemple 2 : Cet exemple désactive la prise en charge des noms d'hôte DNS pour le VPC spécifié.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $false
```

Exemple 3 : Cet exemple active la prise en charge de la résolution DNS pour le VPC spécifié.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $true
```

Exemple 4 : Cet exemple désactive la prise en charge de la résolution DNS pour le VPC spécifié.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $false
```

- Pour plus de détails sur l'API, reportez-vous [ModifyVpcAttribute](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **MonitorInstances** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `MonitorInstances`.

C++

Kit de développement logiciel (SDK) for C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::MonitorInstancesRequest request;
request.AddInstanceIds(instanceId);
request.SetDryRun(true);

auto dry_run_outcome = ec2Client.MonitorInstances(request);
if (dry_run_outcome.IsSuccess()) {
    std::cerr
        << "Failed dry run to enable monitoring on instance. A dry run
should trigger an error."
        <<
        std::endl;
    return false;
}
else if (dry_run_outcome.GetError().GetErrorType()
    != Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
    std::cerr << "Failed dry run to enable monitoring on instance " <<
        instanceId << ": " << dry_run_outcome.GetError().GetMessage()
    <<
        std::endl;
```

```
        return false;
    }

    request.SetDryRun(false);
    auto monitorInstancesOutcome = ec2Client.MonitorInstances(request);
    if (!monitorInstancesOutcome.IsSuccess()) {
        std::cerr << "Failed to enable monitoring on instance " <<
            instanceId << ": " <<
            monitorInstancesOutcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully enabled monitoring on instance " <<
            instanceId << std::endl;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [MonitorInstances](#) à la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Activation de la surveillance détaillée d'une instance

Cet exemple de commande active la surveillance détaillée de l'instance spécifiée.

Commande :

```
aws ec2 monitor-instances --instance-ids i-1234567890abcdef0
```

Sortie :

```
{
  "InstanceMonitorings": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "Monitoring": {
        "State": "pending"
      }
    }
  ]
}
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [MonitorInstances](#) à la section Référence des AWS CLI commandes.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

// Turn on detailed monitoring for the selected instance.
// By default, metrics are sent to Amazon CloudWatch every 5 minutes.
// For a cost you can enable detailed monitoring which sends metrics every
minute.
export const main = async () => {
  const command = new MonitorInstancesCommand({
    InstanceIds: ["INSTANCE_ID"],
  });

  try {
    const { InstanceMonitorings } = await client.send(command);
    const instancesBeingMonitored = InstanceMonitorings.map(
      (im) =>
        ` • Detailed monitoring state for ${im.InstanceId} is
${im.Monitoring.State}.`,
    );
    console.log("Monitoring status:");
    console.log(instancesBeingMonitored.join("\n"));
  } catch (err) {
    console.error(err);
  }
};
```

- Pour plus de détails sur l'API, reportez-vous [MonitorInstances](#) à la section Référence des AWS SDK for JavaScript API.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple permet une surveillance détaillée de l'instance spécifiée.

```
Start-EC2InstanceMonitoring -InstanceId i-12345678
```

Sortie :

```
InstanceId      Monitoring
-----
i-12345678     Amazon.EC2.Model.Monitoring
```

- Pour plus de détails sur l'API, reportez-vous [MonitorInstances](#) à la section Référence des AWS Tools for PowerShell applets de commande.

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
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.

```

- Pour plus de détails sur l'API, reportez-vous [MonitorInstances](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **MoveAddressToVpc** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `MoveAddressToVpc`.

CLI

AWS CLI

Pour déplacer une adresse vers EC2-VPC

Cet exemple déplace l'adresse IP élastique 54.123.4.56 vers la plateforme EC2-VPC.

Commande :

```
aws ec2 move-address-to-vpc --public-ip 54.123.4.56
```

Sortie :

```
{  
  "Status": "MoveInProgress"  
}
```

- Pour plus de détails sur l'API, reportez-vous [MoveAddressToVpc](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple déplace une instance EC2 dont l'adresse IP publique est 12.345.67.89 vers la plate-forme EC2-VPC dans la région USA Est (Virginie du Nord).

```
Move-EC2AddressToVpc -PublicIp 12.345.67.89 -Region us-east-1
```

Exemple 2 : Cet exemple dirige les résultats d'une Get-EC2Instance commande vers l'Move-EC2AddressToVpc applet de commande. La Get-EC2Instance commande obtient une instance spécifiée par son ID d'instance, puis renvoie la propriété d'adresse IP publique de l'instance.

```
(Get-EC2Instance -Instance i-12345678).Instances.PublicIpAddress | Move-  
EC2AddressToVpc
```

- Pour plus de détails sur l'API, reportez-vous [MoveAddressToVpc](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **PurchaseHostReservation** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `PurchaseHostReservation`.

CLI

AWS CLI

Pour acheter une réservation d'hôte dédié

Cet exemple achète l'offre de réservation d'hôte dédié spécifiée pour l'hôte dédié spécifié sur votre compte.

Commande :

```
aws ec2 purchase-host-reservation --offering-id hro-03f707bf363b6b324 --host-id-set h-013abcd2a00cbd123
```

Sortie :

```
{
  "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"
}
```

- Pour plus de détails sur l'API, reportez-vous [PurchaseHostReservation](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple achète l'offre de réservation hro-0c1f23456789d0ab avec des configurations qui correspondent à celles de votre hôte dédié h-01e23f4cd567890f1

```
New-EC2HostReservation -OfferingId hro-0c1f23456789d0ab HostIdSet  
h-01e23f4cd567890f1
```

Sortie :

```
ClientToken      :  
CurrencyCode    :  
Purchase        : {hr-0123f4b5d67bedc89}  
TotalHourlyPrice : 1.307  
TotalUpfrontPrice : 0.000
```

- Pour plus de détails sur l'API, reportez-vous [PurchaseHostReservation](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **PurchaseScheduledInstances** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `PurchaseScheduledInstances`.

CLI

AWS CLI

Pour acheter une instance planifiée

Cet exemple achète une instance planifiée.

Commande :

```
aws ec2 purchase-scheduled-instances --purchase-requests file://purchase-request.json
```

Purchase-Request.json :

```
[
  {
    "PurchaseToken": "eyJ2IjoiMSIsInMiOiJEsImMiOi...",
    "InstanceCount": 1
  }
]
```

Sortie :

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

```
}
```

- Pour plus de détails sur l'API, reportez-vous [PurchaseScheduledInstances](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple achète une instance planifiée.

```
$request = New-Object Amazon.EC2.Model.PurchaseRequest
$request.InstanceCount = 1
$request.PurchaseToken = "eyJ2IjoiMSIsInMiOiJEsImMiOi..."
New-EC2ScheduledInstancePurchase -PurchaseRequest $request
```

Sortie :

```
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
```

- Pour plus de détails sur l'API, reportez-vous [PurchaseScheduledInstances](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **RebootInstances** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `RebootInstances`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Créer et gérer un service résilient](#)

.NET

AWS SDK for .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

    var response = await _amazonEC2.RebootInstancesAsync(request);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine("Instances successfully rebooted.");
    }
    else
    {
        Console.WriteLine("Could not reboot one or more instances.");
    }
}
```

```

    }
}

```

Remplacez le profil par une instance, redémarrez et redémarrez un serveur web.

```

    /// <summary>
    /// Replace the profile associated with a running instance. After the profile
    is replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
    ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate
    with the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association
    for the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
    credsProfileName, string associationId)
    {
        await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
            new ReplaceIamInstanceProfileAssociationRequest()
            {
                AssociationId = associationId,
                IamInstanceProfile = new IamInstanceProfileSpecification()
                {
                    Name = credsProfileName
                }
            });
        // Allow time before resetting.
        Thread.Sleep(25000);
        var instanceReady = false;
        var retries = 5;
        while (retries-- > 0 && !instanceReady)
        {
            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(10000);

            var instancesPaginator =
            _amazonSsm.Paginators.DescribeInstanceInformation(

```

```
        new DescribeInstanceInformationRequest());
    // Get the entire list using the paginator.
    await foreach (var instance in
instancesPaginator.InstanceInformationList)
    {
        instanceReady = instance.InstanceId == instanceId;
        if (instanceReady)
        {
            break;
        }
    }
    Console.WriteLine($"Sending restart command to instance {instanceId}");
    await _amazonSsm.SendCommandAsync(
        new SendCommandRequest()
        {
            InstanceIds = new List<string>() { instanceId },
            DocumentName = "AWS-RunShellScript",
            Parameters = new Dictionary<string, List<string>>()
            {
                {"commands", new List<string>() { "cd / && sudo python3
server.py 80" }}
            }
        });
    Console.WriteLine($"Restarted the web server on instance {instanceId}");
}
```

- Pour plus de détails sur l'API, reportez-vous [RebootInstances](#) à la section Référence des AWS SDK for .NET API.

C++

SDK pour C++

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

Aws::EC2::Model::RebootInstancesRequest request;
request.AddInstanceIds(instanceId);
request.SetDryRun(true);

auto dry_run_outcome = ec2Client.RebootInstances(request);
if (dry_run_outcome.IsSuccess()) {
    std::cerr
        << "Failed dry run to reboot on instance. A dry run should
trigger an error."
        <<
        std::endl;
    return false;
}
else if (dry_run_outcome.GetError().GetErrorType()
    != Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
    std::cout << "Failed dry run to reboot instance " << instanceId << ": "
        << dry_run_outcome.GetError().GetMessage() << std::endl;
    return false;
}

request.SetDryRun(false);
auto outcome = ec2Client.RebootInstances(request);
if (!outcome.IsSuccess()) {
    std::cout << "Failed to reboot instance " << instanceId << ": " <<
        outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Successfully rebooted instance " << instanceId <<
        std::endl;
}
}
```

- Pour plus de détails sur l'API, reportez-vous [RebootInstances](#) à la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Redémarrage d'une instance Amazon EC2

Cet exemple redémarre l'instance spécifiée. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 reboot-instances --instance-ids i-1234567890abcdef5
```

Pour plus d'informations, veuillez consulter la rubrique Redémarrer votre instance dans le Guide de l'utilisateur Amazon Elastic Compute Cloud.

- Pour plus de détails sur l'API, reportez-vous [RebootInstances](#) à la section Référence des AWS CLI commandes.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

export const main = async () => {
  const command = new RebootInstancesCommand({
    InstanceIds: ["INSTANCE_ID"],
  });

  try {
    await client.send(command);
    console.log("Instance rebooted successfully.");
  } catch (err) {
    console.error(err);
  }
};
```

- Pour plus de détails sur l'API, reportez-vous [RebootInstances](#) à la section Référence des AWS SDK for JavaScript API.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple redémarre l'instance spécifiée.

```
Restart-EC2Instance -InstanceId i-12345678
```

- Pour plus de détails sur l'API, reportez-vous [RebootInstances](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Python

SDK pour Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

    def __init__(
        self,
        resource_prefix,
        inst_type,
        ami_param,
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    ):
        """
```

```

        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is
            created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        self.launch_template_name = f"{resource_prefix}-template"
        self.group_name = f"{resource_prefix}-group"
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"
        self.key_pair_name = f"{resource_prefix}-key-pair"

    def replace_instance_profile(
        self, instance_id, new_instance_profile_name, profile_association_id
    ):
        """
        Replaces the profile associated with a running instance. After the
        profile is
            replaced, the instance is rebooted to ensure that it uses the new
        profile. When
            the instance is ready, Systems Manager is used to restart the Python web
        server.

        :param instance_id: The ID of the instance to update.
        :param new_instance_profile_name: The name of the new profile to
        associate with
            the specified instance.

```

```

        :param profile_association_id: The ID of the existing profile association
for the
                                instance.
    """
    try:
        self.ec2_client.replace_iam_instance_profile_association(
            IamInstanceProfile={"Name": new_instance_profile_name},
            AssociationId=profile_association_id,
        )
        log.info(
            "Replaced instance profile for association %s with profile %s.",
            profile_association_id,
            new_instance_profile_name,
        )
        time.sleep(5)
        inst_ready = False
        tries = 0
        while not inst_ready:
            if tries % 6 == 0:
                self.ec2_client.reboot_instances(InstanceIds=[instance_id])
                log.info(
                    "Rebooting instance %s and waiting for it to be
ready.",
                                instance_id,
                )
                tries += 1
                time.sleep(10)
                response = self.ssm_client.describe_instance_information()
                for info in response["InstanceInformationList"]:
                    if info["InstanceId"] == instance_id:
                        inst_ready = True
            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
            log.info("Restarted the Python web server on instance %s.",
instance_id)
        except ClientError as err:
            raise AutoScalerError(
                f"Couldn't replace instance profile for association
{profile_association_id}: {err}"
            )

```

- Pour plus de détails sur l'API, consultez [RebootInstances](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Rust

SDK pour Rust

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
async fn reboot_instance(client: &Client, id: &str) -> Result<(), Error> {
    println!("Rebooting instance.");

    client.reboot_instances().instance_ids(id).send().await?;

    client
        .wait_until_instance_stopped()
        .instance_ids(id)
        .wait(Duration::from_secs(60))
        .await?;
    let wait_status_ok = client
        .wait_until_instance_status_ok()
        .instance_ids(id)
        .wait(Duration::from_secs(60))
        .await;

    match wait_status_ok {
        Ok(_) => println!("Rebooted instance {id}, it is started with status
OK."),
        Err(err) => return Err(err.into()),
    }

    Ok(())
}
```

- Pour plus de détails sur l'API, voir [RebootInstances](#) la section de référence de l'API AWS SDK for Rust.

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
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.
```

- Pour plus de détails sur l'API, reportez-vous [RebootInstances](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **RegisterImage** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `RegisterImage`.

CLI

AWS CLI

Exemple 1 : pour enregistrer une AMI à l'aide d'un fichier manifeste

L'`register-image`exemple suivant enregistre une AMI à l'aide du fichier manifeste spécifié dans Amazon S3.

```
aws ec2 register-image \  
  --name my-image \  
  --image-location my-s3-bucket/myimage/image.manifest.xml
```

Sortie :

```
{  
  "ImageId": "ami-1234567890EXAMPLE"  
}
```

Pour plus d'informations, veuillez consulter la rubrique [Amazon Machine Images \(AMI\)](#) dans le Guide de l'utilisateur Amazon EC2.

Exemple 2 : pour enregistrer une AMI à l'aide d'un instantané d'un périphérique racine

L'`register-image` suivant enregistre une AMI en utilisant l'instantané spécifié d'un volume racine EBS en tant que périphérique `/dev/xvda`. Le mappage des périphériques en mode bloc inclut également un volume EBS vide de 100 GiB en tant que périphérique `/dev/xvdf`

```
aws ec2 register-image \  
  --name my-image \  
  --root-device-name /dev/xvda \  
  --block-device-mappings DeviceName=/dev/  
xvda,Ebs={SnapshotId=snap-0db2cf683925d191f} DeviceName=/dev/  
xvdf,Ebs={VolumeSize=100}
```

Sortie :

```
{  
  "ImageId": "ami-1a2b3c4d5eEXAMPLE"  
}
```

Pour plus d'informations, veuillez consulter la rubrique [Amazon Machine Images \(AMI\)](#) dans le Guide de l'utilisateur Amazon EC2.

- Pour plus de détails sur l'API, reportez-vous [RegisterImage](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple enregistre une AMI à l'aide du fichier manifeste spécifié dans Amazon S3.

```
Register-EC2Image -ImageLocation my-s3-bucket/my-web-server-ami/  
image.manifest.xml -Name my-web-server-ami
```

- Pour plus de détails sur l'API, reportez-vous [RegisterImage](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation `RejectVpcPeeringConnection` avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `RejectVpcPeeringConnection`.

CLI

AWS CLI

Pour rejeter une connexion d'appairage VPC

Cet exemple rejette la demande de connexion d'appairage VPC spécifiée.

Commande :

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

Sortie :

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

- Pour plus de détails sur l'API, reportez-vous [RejectVpcPeeringConnection](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : L'exemple ci-dessus refuse la demande d'identifiant de demande `VpcPeering pcx-01a2b3ce45fe67eb8`

```
Deny-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-01a2b3ce45fe67eb8
```

- Pour plus de détails sur l'API, reportez-vous [RejectVpcPeeringConnection](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **ReleaseAddress** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ReleaseAddress`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Démarrer avec les instances](#)

.NET

AWS SDK for .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Release an Elastic IP address.
/// </summary>
/// <param name="allocationId">The allocation Id of the Elastic IP address.</
param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> ReleaseAddress(string allocationId)
{
    var request = new ReleaseAddressRequest
    {
        AllocationId = allocationId
    };

    var response = await _amazonEC2.ReleaseAddressAsync(request);
    return response.HttpStatusCode == HttpStatusCode.OK;
}
```

- Pour plus de détails sur l'API, reportez-vous [ReleaseAddress](#) à la section Référence des AWS SDK for .NET API.

Bash

AWS CLI avec le script Bash

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
#####
# 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
}

```

Fonctions utilitaires utilisées dans cet exemple.

```

#####
# 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
}

```

- Pour plus de détails sur l'API, reportez-vous [ReleaseAddress](#) à la section Référence des AWS CLI commandes.

C++

SDK pour C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
Aws::EC2::EC2Client ec2(clientConfiguration);

Aws::EC2::Model::ReleaseAddressRequest request;
request.SetAllocationId(allocationID);

auto outcome = ec2.ReleaseAddress(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Failed to release Elastic IP address " <<
        allocationID << ":" << outcome.GetError().GetMessage() <<
        std::endl;
}
else {
    std::cout << "Successfully released Elastic IP address " <<
        allocationID << std::endl;
}
```

- Pour plus de détails sur l'API, reportez-vous [ReleaseAddress](#) à la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Libération d'une adresse IP Elastic pour EC2-Classic

Cet exemple libère une adresse IP Elastic pour pouvoir l'utiliser avec des instances dans EC2-Classic. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 release-address --public-ip 198.51.100.0
```

Libération d'une adresse IP Elastic pour EC2-VPC

Cet exemple libère une adresse IP Elastic pour pouvoir l'utiliser avec des instances dans un VPC. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 release-address --allocation-id eipalloc-64d5890a
```

- Pour plus de détails sur l'API, reportez-vous [ReleaseAddress](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void releaseEC2Address(Ec2Client ec2, String allocId) {
    try {
        ReleaseAddressRequest request = ReleaseAddressRequest.builder()
            .allocationId(allocId)
            .build();

        ec2.releaseAddress(request);
        System.out.println("Successfully released Elastic IP address " +
allocId);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ReleaseAddress](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

export const main = async () => {
  const command = new ReleaseAddressCommand({
    // You can also use PublicIp, but that is for EC2 classic which is being
    // retired.
    AllocationId: "ALLOCATION_ID",
  });

  try {
    await client.send(command);
    console.log("Successfully released address.");
  } catch (err) {
    console.error(err);
  }
};
```

- Pour plus de détails sur l'API, reportez-vous [ReleaseAddress](#) à la section Référence des AWS SDK for JavaScript API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun releaseEC2AddressSc(allocId: String?) {
    val request =
        ReleaseAddressRequest {
            allocationId = allocId
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.releaseAddress(request)
        println("Successfully released Elastic IP address $allocId")
    }
}
```

- Pour plus de détails sur l'API, consultez [ReleaseAddress](#) la section AWS SDK pour la référence de l'API Kotlin.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple libère l'adresse IP élastique spécifiée pour les instances d'un VPC.

```
Remove-EC2Address -AllocationId eipalloc-12345678 -Force
```

Exemple 2 : Cet exemple libère l'adresse IP élastique spécifiée pour les instances dans EC2-Classic.

```
Remove-EC2Address -PublicIp 198.51.100.2 -Force
```

- Pour plus de détails sur l'API, reportez-vous [ReleaseAddress](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Python

SDK pour Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

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

    def release(self):
        """
        Releases an Elastic IP address. After the Elastic IP address is released,
        it can no longer be used.
        """
```

```
if self.elastic_ip is None:
    logger.info("No Elastic IP to release.")
    return

try:
    self.elastic_ip.release()
except ClientError as err:
    logger.error(
        "Couldn't release Elastic IP address %s. Here's why: %s: %s",
        self.elastic_ip.allocation_id,
        err.response["Error"]["Code"],
        err.response["Error"]["Message"],
    )
    raise
```

- Pour plus de détails sur l'API, consultez [ReleaseAddress](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
# Releases an Elastic IP address from an
# Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - An Amazon EC2 instance with an associated Elastic IP address.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param allocation_id [String] The ID of the allocation corresponding to
#   the Elastic IP address.
# @return [Boolean] true if the Elastic IP address was released;
```

```

# otherwise, false.
# @example
# exit 1 unless elastic_ip_address_released?(
#   Aws::EC2::Client.new(region: 'us-west-2'),
#   'eipalloc-04452e528a66279EX'
# )
def elastic_ip_address_released?(ec2_client, allocation_id)
  ec2_client.release_address(allocation_id: allocation_id)
  return true
rescue StandardError => e
  puts("Error releasing Elastic IP address: #{e.message}")
  return false
end

```

- Pour plus de détails sur l'API, reportez-vous [ReleaseAddress](#) à la section Référence des AWS SDK for Ruby API.

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

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.

```

- Pour plus de détails sur l'API, reportez-vous [ReleaseAddress](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **ReleaseHosts** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ReleaseHosts`.

CLI

AWS CLI

Pour libérer un hôte dédié de votre compte

Pour libérer un hôte dédié de votre compte. Les instances présentes sur l'hôte doivent être arrêtées ou résiliées avant que l'hôte ne puisse être libéré.

Commande :

```
aws ec2 release-hosts --host-id=h-0029d6e3cacf1b3da
```

Sortie :

```
{
  "Successful": [
    "h-0029d6e3cacf1b3da"
  ],
  "Unsuccessful": []
}
```

- Pour plus de détails sur l'API, reportez-vous [ReleaseHosts](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple libère l'ID d'hôte donné h-0badafd1dcb2f3456

```
Remove-EC2Host -HostId h-0badafd1dcb2f3456
```

Sortie :

```
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} {}
```

- Pour plus de détails sur l'API, reportez-vous [ReleaseHosts](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **ReplaceIamInstanceProfileAssociation** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ReplaceIamInstanceProfileAssociation`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Créer et gérer un service résilient](#)

.NET

AWS SDK for .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Replace the profile associated with a running instance. After the profile
is replaced, the instance
/// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
/// used to restart the Python web server.
/// </summary>
/// <param name="instanceId">The Id of the instance to update.</param>
/// <param name="credsProfileName">The name of the new profile to associate
with the specified instance.</param>
/// <param name="associationId">The Id of the existing profile association
for the instance.</param>
/// <returns>Async task.</returns>
public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
{
    await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
        new ReplaceIamInstanceProfileAssociationRequest()
        {
            AssociationId = associationId,
            IamInstanceProfile = new IamInstanceProfileSpecification()
            {
                Name = credsProfileName
            }
        });
    // Allow time before resetting.
    Thread.Sleep(25000);
    var instanceReady = false;
    var retries = 5;
    while (retries-- > 0 && !instanceReady)
    {
        await _amazonEc2.RebootInstancesAsync(
```

```
        new RebootInstancesRequest(new List<string>() { instanceId }));
        Thread.Sleep(10000);

        var instancesPaginator =
            _amazonSsm.Paginators.DescribeInstanceInformation(
                new DescribeInstanceInformationRequest());
        // Get the entire list using the paginator.
        await foreach (var instance in
            instancesPaginator.InstanceInformationList)
        {
            instanceReady = instance.InstanceId == instanceId;
            if (instanceReady)
            {
                break;
            }
        }
        Console.WriteLine($"Sending restart command to instance {instanceId}");
        await _amazonSsm.SendCommandAsync(
            new SendCommandRequest()
            {
                InstanceIds = new List<string>() { instanceId },
                DocumentName = "AWS-RunShellScript",
                Parameters = new Dictionary<string, List<string>>()
                {
                    {"commands", new List<string>() { "cd / && sudo python3
server.py 80" }}
                }
            });
        Console.WriteLine($"Restarted the web server on instance {instanceId}");
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ReplacelamInstanceProfileAssociation](#) à la section Référence des AWS SDK for .NET API.

CLI

AWS CLI

Remplacement d'un profil d'instance IAM pour une instance

Cet exemple remplace le profil d'instance IAM représenté par l'association `iip-assoc-060bae234aac2e7fa` par le profil d'instance IAM nommé `AdminRole`.

```
aws ec2 replace-iam-instance-profile-association \  
  --iam-instance-profile Name=AdminRole \  
  --association-id iip-assoc-060bae234aac2e7fa
```

Sortie :

```
{  
  "IamInstanceProfileAssociation": {  
    "InstanceId": "i-087711ddaf98f9489",  
    "State": "associating",  
    "AssociationId": "iip-assoc-0b215292fab192820",  
    "IamInstanceProfile": {  
      "Id": "AIPAJLNLDX3AMYZNWYYAY",  
      "Arn": "arn:aws:iam::123456789012:instance-profile/AdminRole"  
    }  
  }  
}
```

- Pour plus de détails sur l'API, reportez-vous [ReplacelamInstanceProfileAssociation](#) à la section Référence des AWS CLI commandes.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>  
  ec2Client.send(  
    new ReplaceIamInstanceProfileAssociationCommand({  
      AssociationId: state.instanceProfileAssociationId,  
      IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },  
    })),
```

```
    ),  
    );
```

- Pour plus de détails sur l'API, reportez-vous [ReplacelamInstanceProfileAssociation](#) à la section Référence des AWS SDK for JavaScript API.

Python

SDK pour Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Cet exemple remplace le profil d'instance d'une instance en cours d'exécution, redémarre l'instance et envoie une commande à l'instance après son démarrage.

```
class AutoScaler:  
    """  
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.  
    """  
  
    def __init__(  
        self,  
        resource_prefix,  
        inst_type,  
        ami_param,  
        autoscaling_client,  
        ec2_client,  
        ssm_client,  
        iam_client,  
    ):  
        """  
        :param resource_prefix: The prefix for naming AWS resources that are  
        created by this class.  
        :param inst_type: The type of EC2 instance to create, such as t3.micro.  
        :param ami_param: The Systems Manager parameter used to look up the AMI  
        that is
```

```

        created.
    :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
    :param ec2_client: A Boto3 EC2 client.
    :param ssm_client: A Boto3 Systems Manager client.
    :param iam_client: A Boto3 IAM client.
    """
    self.inst_type = inst_type
    self.ami_param = ami_param
    self.autoscaling_client = autoscaling_client
    self.ec2_client = ec2_client
    self.ssm_client = ssm_client
    self.iam_client = iam_client
    self.launch_template_name = f"{resource_prefix}-template"
    self.group_name = f"{resource_prefix}-group"
    self.instance_policy_name = f"{resource_prefix}-pol"
    self.instance_role_name = f"{resource_prefix}-role"
    self.instance_profile_name = f"{resource_prefix}-prof"
    self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
    self.bad_creds_role_name = f"{resource_prefix}-bc-role"
    self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"
    self.key_pair_name = f"{resource_prefix}-key-pair"

    def replace_instance_profile(
        self, instance_id, new_instance_profile_name, profile_association_id
    ):
        """
        Replaces the profile associated with a running instance. After the
        profile is
        replaced, the instance is rebooted to ensure that it uses the new
        profile. When
        the instance is ready, Systems Manager is used to restart the Python web
        server.

        :param instance_id: The ID of the instance to update.
        :param new_instance_profile_name: The name of the new profile to
        associate with
            the specified instance.
        :param profile_association_id: The ID of the existing profile association
        for the
            instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(

```

```
        IamInstanceProfile={"Name": new_instance_profile_name},
        AssociationId=profile_association_id,
    )
    log.info(
        "Replaced instance profile for association %s with profile %s.",
        profile_association_id,
        new_instance_profile_name,
    )
    time.sleep(5)
    inst_ready = False
    tries = 0
    while not inst_ready:
        if tries % 6 == 0:
            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info(
                "Rebooting instance %s and waiting for it to be
ready.",
                instance_id,
            )
            tries += 1
            time.sleep(10)
            response = self.ssm_client.describe_instance_information()
            for info in response["InstanceInformationList"]:
                if info["InstanceId"] == instance_id:
                    inst_ready = True
            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
            log.info("Restarted the Python web server on instance %s.",
instance_id)
        except ClientError as err:
            raise AutoScalerError(
                f"Couldn't replace instance profile for association
{profile_association_id}: {err}")
    )
```

- Pour plus de détails sur l'API, consultez [ReplacelamInstanceProfileAssociation](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation `ReplaceNetworkAclAssociation` avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ReplaceNetworkAclAssociation`.

CLI

AWS CLI

Pour remplacer l'ACL réseau associée à un sous-réseau

Cet exemple associe l'ACL réseau spécifiée au sous-réseau pour l'association ACL réseau spécifiée.

Commande :

```
aws ec2 replace-network-acl-association --association-id aclassoc-e5b95c8c --network-acl-id acl-5fb85d36
```

Sortie :

```
{
  "NewAssociationId": "aclassoc-3999875b"
}
```

- Pour plus de détails sur l'API, reportez-vous [ReplaceNetworkAclAssociation](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple associe l'ACL réseau spécifiée au sous-réseau pour l'association ACL réseau spécifiée.

```
Set-EC2NetworkAclAssociation -NetworkAclId acl-12345678 -AssociationId  
aclassoc-1a2b3c4d
```

Sortie :

```
aclassoc-87654321
```

- Pour plus de détails sur l'API, reportez-vous [ReplaceNetworkAclAssociation](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **ReplaceNetworkAclEntry** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ReplaceNetworkAclEntry`.

CLI

AWS CLI

Pour remplacer une entrée ACL réseau

Cet exemple remplace une entrée pour l'ACL réseau spécifiée. La nouvelle règle 100 autorise le trafic entrant du 203.0.113.12/24 sur le port UDP 53 (DNS) vers n'importe quel sous-réseau associé.

Commande :

```
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
```

- Pour plus de détails sur l'API, reportez-vous [ReplaceNetworkAclEntry](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple remplace l'entrée spécifiée pour l'ACL réseau spécifiée. La nouvelle règle autorise le trafic entrant depuis l'adresse spécifiée vers n'importe quel sous-réseau associé.

```
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
```

- Pour plus de détails sur l'API, reportez-vous [ReplaceNetworkAclEntry](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **ReplaceRoute** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ReplaceRoute`.

CLI

AWS CLI

Pour remplacer un itinéraire

Cet exemple remplace l'itinéraire spécifié dans la table de routage spécifiée. La nouvelle route correspond au CIDR spécifié et envoie le trafic vers la passerelle privée virtuelle spécifiée. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 replace-route --route-table-id rtb-22574640 --destination-cidr-block  
10.0.0.0/16 --gateway-id vgw-9a4cacf3
```

- Pour plus de détails sur l'API, reportez-vous [ReplaceRoute](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple remplace l'itinéraire spécifié par la table de routage spécifiée. La nouvelle route envoie le trafic spécifié à la passerelle privée virtuelle spécifiée.

```
Set-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 10.0.0.0/24 -  
GatewayId vgw-1a2b3c4d
```

- Pour plus de détails sur l'API, reportez-vous [ReplaceRoute](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **ReplaceRouteTableAssociation** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ReplaceRouteTableAssociation`.

CLI

AWS CLI

Pour remplacer la table de routage associée à un sous-réseau

Cet exemple associe la table de routage spécifiée au sous-réseau pour l'association de table de routage spécifiée.

Commande :

```
aws ec2 replace-route-table-association --association-id rtbassoc-781d0d1a --  
route-table-id rtb-22574640
```

Sortie :

```
{  
  "NewAssociationId": "rtbassoc-3a1f0f58"
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [ReplaceRouteTableAssociation](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple associe la table de routage spécifiée au sous-réseau pour l'association de table de routage spécifiée.

```
Set-EC2RouteTableAssociation -RouteTableId rtb-1a2b3c4d -AssociationId  
rtbassoc-12345678
```

Sortie :

```
rtbassoc-87654321
```

- Pour plus de détails sur l'API, reportez-vous [ReplaceRouteTableAssociation](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **ReportInstanceStatus** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ReportInstanceStatus`.

CLI

AWS CLI

Pour signaler des commentaires sur le statut d'une instance

Cet exemple de commande indique le statut de l'instance spécifiée.

Commande :

```
aws ec2 report-instance-status --instances i-1234567890abcdef0 --status impaired
--reason-codes unresponsive
```

- Pour plus de détails sur l'API, reportez-vous [ReportInstanceStatus](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple indique le statut de l'instance spécifiée.

```
Send-EC2InstanceStatus -Instance i-12345678 -Status impaired -ReasonCode
unresponsive
```

- Pour plus de détails sur l'API, reportez-vous [ReportInstanceStatus](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **RequestSpotFleet** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `RequestSpotFleet`.

CLI

AWS CLI

Pour demander une flotte Spot dans le sous-réseau au prix le plus bas

Cet exemple de commande crée une demande de flotte Spot avec deux spécifications de lancement qui ne diffèrent que selon le sous-réseau. La flotte Spot lance les instances dans le sous-réseau spécifié au prix le plus bas. Si les instances sont lancées dans un VPC par défaut, elles reçoivent une adresse IP publique par défaut. Si les instances sont lancées sur un VPC personnalisé, elles ne reçoivent pas d'adresse IP publique par défaut.

Notez que vous ne pouvez pas spécifier différents sous-réseaux provenant de la même zone de disponibilité dans une demande de flotte Spot.

Commande :

```
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"
      }
    }
  ]
}
```

Sortie :

```
{
  "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE"
}
```

Pour demander une flotte Spot dans la zone de disponibilité au prix le plus bas

Cet exemple de commande crée une demande de flotte Spot avec deux spécifications de lancement qui ne diffèrent que selon la zone de disponibilité. La flotte Spot lance les instances dans la zone de disponibilité spécifiée au prix le plus bas. Si votre compte prend uniquement en charge l'EC2-VPC, Amazon EC2 lance les instances Spot dans le sous-réseau par défaut

de la zone de disponibilité. Si votre compte prend en charge EC2-Classic, Amazon EC2 lance les instances dans EC2-Classic dans la zone de disponibilité.

Commande :

```
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"
      }
    }
  ]
}
```

Pour lancer des instances Spot dans un sous-réseau et leur attribuer des adresses IP publiques

Cet exemple de commande attribue des adresses publiques aux instances lancées dans un VPC autre que celui par défaut. Notez que lorsque vous spécifiez une interface réseau, vous devez inclure l'ID du sous-réseau et l'ID du groupe de sécurité à l'aide de l'interface réseau.

Commande :

```
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"
      }
    }
  ]
}
```

Pour demander une flotte Spot en utilisant la stratégie d'allocation diversifiée

Cet exemple de commande crée une demande de flotte Spot qui lance 30 instances en utilisant la stratégie d'allocation diversifiée. Les spécifications de lancement varient selon le type d'instance. La flotte Spot répartit les instances selon les spécifications de lancement, de sorte qu'il existe 10 instances de chaque type.

Commande :

```
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"
  }
]
```

Pour plus d'informations, consultez la section [Spot Fleet Requests](#) dans le guide de l'utilisateur d'Amazon Elastic Compute Cloud.

- Pour plus de détails sur l'API, reportez-vous [RequestSpotFleet](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple crée une demande de parc Spot dans la zone de disponibilité avec le prix le plus bas pour le type d'instance spécifié. Si votre compte prend uniquement en charge l'EC2-VPC, le parc Spot lance les instances dans la zone de disponibilité la moins chère dotée d'un sous-réseau par défaut. Si votre compte prend en charge EC2-Classique, le parc Spot lance les instances dans EC2-Classique dans la zone de disponibilité la moins chère. Notez que le prix que vous payez ne dépassera pas le prix spot spécifié pour la demande.

```
$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
```

- Pour plus de détails sur l'API, reportez-vous [RequestSpotFleet](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **RequestSpotInstances** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `RequestSpotInstances`.

CLI

AWS CLI

Pour demander des instances Spot

Cet exemple de commande crée une demande d'instance ponctuelle unique pour cinq instances dans la zone de disponibilité spécifiée. Si votre compte prend uniquement en charge EC2-VPC, Amazon EC2 lance les instances dans le sous-réseau par défaut de la zone de disponibilité spécifiée. Si votre compte prend en charge EC2-Classical, Amazon EC2 lance les instances dans EC2-Classical dans la zone de disponibilité spécifiée.

Commande :

```
aws ec2 request-spot-instances --spot-price "0.03" --instance-count 5 --type
"one-time" --launch-specification file://specification.json
```

Spécification.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"
}
}

```

Sortie :

```

{
  "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"
},
...
]
}

```

Cet exemple de commande crée une demande d'instance Spot unique pour cinq instances dans le sous-réseau spécifié. Amazon EC2 lance les instances dans le sous-réseau spécifié. Si le VPC n'est pas un VPC par défaut, les instances ne reçoivent pas d'adresse IP publique par défaut.

Commande :

```
aws ec2 request-spot-instances --spot-price "0.050" --instance-count 5 --type
"one-time" --launch-specification file://specification.json
```

Spécification.json :

```

{
  "ImageId": "ami-1a2b3c4d",
  "SecurityGroupIds": [ "sg-1a2b3c4d" ],
  "InstanceType": "m3.medium",
  "SubnetId": "subnet-1a2b3c4d",
  "IamInstanceProfile": {
    "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
  }
}

```

Sortie :

```

{
  "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"
},
...
]
```

Cet exemple attribue une adresse IP publique aux instances Spot que vous lancez dans un VPC autre que celui par défaut. Notez que lorsque vous spécifiez une interface réseau, vous devez inclure l'ID du sous-réseau et l'ID du groupe de sécurité à l'aide de l'interface réseau.

Commande :

```
aws ec2 request-spot-instances --spot-price "0.050" --instance-count 1 --type
"one-time" --launch-specification file://specification.json
```

Spécification.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"
  }
}
```

- Pour plus de détails sur l'API, reportez-vous [RequestSpotInstances](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple demande une instance Spot unique dans le sous-réseau spécifié. Notez que le groupe de sécurité doit être créé pour le VPC qui contient le sous-réseau spécifié, et qu'il doit être spécifié par ID à l'aide de l'interface réseau. Lorsque vous spécifiez une interface réseau, vous devez inclure l'ID de sous-réseau à l'aide de l'interface réseau.

```
$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
```

Sortie :

```
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
```

- Pour plus de détails sur l'API, reportez-vous [RequestSpotInstances](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **ResetImageAttribute** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ResetImageAttribute`.

CLI

AWS CLI

Pour réinitialiser l'attribut `LaunchPermission`

Cet exemple rétablit la valeur par défaut de l'`launchPermission` attribut de l'AMI spécifiée. Par défaut, les AMI sont privées. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 reset-image-attribute --image-id ami-5731123e --attribute
  launchPermission
```

- Pour plus de détails sur l'API, reportez-vous [ResetImageAttribute](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple rétablit la valeur par défaut de l'attribut « LaunchPermission ». Par défaut, les AMI sont privées.

```
Reset-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

- Pour plus de détails sur l'API, reportez-vous [ResetImageAttribute](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **ResetInstanceAttribute** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ResetInstanceAttribute`.

CLI

AWS CLI

Pour réinitialiser l' `sourceDestCheck` attribut

Cet exemple réinitialise l'`sourceDestCheck` attribut de l'instance spécifiée. L'instance doit se trouver dans un VPC. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 reset-instance-attribute --instance-id i-1234567890abcdef0 --attribute
  sourceDestCheck
```

Pour réinitialiser l'attribut du noyau

Cet exemple réinitialise l'`kernel`attribut de l'instance spécifiée. L'instance doit être dans l'état `stopped`. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 reset-instance-attribute --instance-id i-1234567890abcdef0 --attribute kernel
```

Pour réinitialiser l'attribut ramdisk

Cet exemple réinitialise l'`ramdisk`attribut de l'instance spécifiée. L'instance doit être dans l'état `stopped`. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 reset-instance-attribute --instance-id i-1234567890abcdef0 --attribute ramdisk
```

- Pour plus de détails sur l'API, reportez-vous [ResetInstanceAttribute](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple réinitialise l'attribut « `sriovNetSupport` » pour l'instance spécifiée.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

Exemple 2 : Cet exemple réinitialise l'attribut « `EBSOptimized` » pour l'instance spécifiée.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

Exemple 3 : Cet exemple réinitialise l'attribut « `sourceDestCheck` » pour l'instance spécifiée.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sourceDestCheck
```

Exemple 4 : Cet exemple réinitialise l'attribut « `disableApiTermination` » pour l'instance spécifiée.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute  
disableApiTermination
```

Exemple 5 : Cet exemple réinitialise l'attribut « `instanceInitiatedShutdown Comportement` » pour l'instance spécifiée.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute  
instanceInitiatedShutdownBehavior
```

- Pour plus de détails sur l'API, reportez-vous [ResetInstanceAttribute](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **ResetNetworkInterfaceAttribute** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ResetNetworkInterfaceAttribute`.

CLI

AWS CLI

Pour réinitialiser un attribut d'interface réseau

L'exemple suivant réinitialise la valeur de l'attribut de vérification source/destination sur `true`

```
aws ec2 reset-network-interface-attribute \  
  --network-interface-id eni-686ea200 \  
  --source-dest-check
```

Cette commande ne produit aucun résultat.

- Pour plus de détails sur l'API, reportez-vous [ResetNetworkInterfaceAttribute](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple réinitialise la vérification source/destination pour l'interface réseau spécifiée.

```
Reset-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -  
SourceDestCheck
```

- Pour plus de détails sur l'API, reportez-vous [ResetNetworkInterfaceAttribute](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **ResetSnapshotAttribute** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `ResetSnapshotAttribute`.

CLI

AWS CLI

Pour réinitialiser un attribut de capture d'écran

Cet exemple réinitialise les autorisations de création de volume pour les instantanés `snap-1234567890abcdef0`. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

```
aws ec2 reset-snapshot-attribute --snapshot-id snap-1234567890abcdef0 --attribute  
createVolumePermission
```

- Pour plus de détails sur l'API, reportez-vous [ResetSnapshotAttribute](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple réinitialise l'attribut spécifié pour le cliché spécifié.

```
Reset-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute  
CreateVolumePermission
```

- Pour plus de détails sur l'API, reportez-vous [ResetSnapshotAttribute](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **RevokeSecurityGroupEgress** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `RevokeSecurityGroupEgress`.

CLI

AWS CLI

Exemple 1 : pour supprimer la règle qui autorise le trafic sortant vers une plage d'adresses spécifique

L'`revoke-security-group-egress` exemple de commande suivant supprime la règle qui autorise l'accès aux plages d'adresses spécifiées sur le port TCP 80.

```
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}]}]
```

Cette commande ne produit aucun résultat.

Pour plus d'informations, consultez [la section Groupes de sécurité](#) dans le guide de l'utilisateur Amazon EC2.

Exemple 2 : pour supprimer la règle qui autorise le trafic sortant vers un groupe de sécurité spécifique

L'`revoke-security-group-egress` de commande suivant supprime la règle qui accorde l'accès au groupe de sécurité spécifié sur le port TCP 80.

```
aws ec2 revoke-security-group-egress \  
  --group-id sg-026c12253ce15eff7 \  
  --ip-permissions '[{"IpProtocol": "tcp", "FromPort": 443, "ToPort":  
  443, "UserIdGroupPairs": [{"GroupId": "sg-06df23a01ff2df86d"}]}]'
```

Cette commande ne produit aucun résultat.

Pour plus d'informations, consultez [la section Groupes de sécurité](#) dans le guide de l'utilisateur Amazon EC2.

- Pour plus de détails sur l'API, reportez-vous [RevokeSecurityGroupEgress](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la règle pour le groupe de sécurité spécifié pour EC2-VPC. Cela révoque l'accès à la plage d'adresses IP spécifiée sur le port TCP 80. La syntaxe utilisée dans cet exemple nécessite PowerShell la version 3 ou supérieure.

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

Exemple 2 : avec PowerShell la version 2, vous devez utiliser `New-Object` pour créer l'`IpPermission` objet.

```
$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
```

Exemple 3 : Cet exemple révoque l'accès au groupe de sécurité source spécifié sur le port TCP 80.

```
$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 } )
```

- Pour plus de détails sur l'API, reportez-vous [RevokeSecurityGroupEgress](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **RevokeSecurityGroupIngress** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `RevokeSecurityGroupIngress`.

CLI

AWS CLI

Exemple 1 : pour supprimer une règle d'un groupe de sécurité

L'`revoke-security-group-ingress` exemple suivant supprime l'accès au port TCP 22 pour la plage d'`203.0.113.0/24` adresses du groupe de sécurité spécifié pour un VPC par défaut.

```
aws ec2 revoke-security-group-ingress \
  --group-name mySecurityGroup
  --protocol tcp \
  --port 22 \
  --cidr 203.0.113.0/24
```

Cette commande ne produit aucune sortie en cas de succès.

Pour plus d'informations, consultez [la section Groupes de sécurité](#) dans le guide de l'utilisateur Amazon EC2.

Exemple 2 : pour supprimer une règle à l'aide des autorisations IP définies

L'`revoke-security-group-ingress` suivant utilise le `ip-permissions` paramètre pour supprimer une règle entrante qui autorise le message ICMP Destination Unreachable: Fragmentation Needed and Don't Fragment was Set (type 3, code 4).

```
aws ec2 revoke-security-group-ingress \  
  --group-id sg-026c12253ce15eff7 \  
  --ip-permissions  
  IpProtocol=icmp,FromPort=3,ToPort=4,IpRanges=[{CidrIp=0.0.0.0/0}]
```

Cette commande ne produit aucune sortie en cas de succès.

Pour plus d'informations, consultez [la section Groupes de sécurité](#) dans le guide de l'utilisateur Amazon EC2.

- Pour plus de détails sur l'API, reportez-vous [RevokeSecurityGroupIngress](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple révoque l'accès au port TCP 22 à partir de la plage d'adresses spécifiée pour le groupe de sécurité spécifié pour EC2-VPC. Notez que vous devez identifier les groupes de sécurité pour EC2-VPC à l'aide de l'ID du groupe de sécurité et non du nom du groupe de sécurité. La syntaxe utilisée dans cet exemple nécessite PowerShell la version 3 ou supérieure.

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22";  
  IpRanges="203.0.113.0/24" }  
Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

Exemple 2 : avec PowerShell la version 2, vous devez utiliser `New-Object` pour créer l'`IpPermission` objet.

```
$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
```

Exemple 3 : Cet exemple révoque l'accès au port TCP 22 à partir de la plage d'adresses spécifiée pour le groupe de sécurité spécifié pour EC2-Classique. La syntaxe utilisée dans cet exemple nécessite PowerShell la version 3 ou supérieure.

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22";
  IpRanges="203.0.113.0/24" }

Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

Exemple 4 : Avec PowerShell la version 2, vous devez utiliser `New-Object` pour créer l'`IpPermission` objet.

```
$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
```

- Pour plus de détails sur l'API, reportez-vous [RevokeSecurityGroupIngress](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **RunInstances** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `RunInstances`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Démarrer avec les instances](#)

.NET

AWS SDK for .NET

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Create and run an EC2 instance.
/// </summary>
/// <param name="ImageId">The image Id of the image used as a basis for the
/// EC2 instance.</param>
/// <param name="instanceType">The instance type of the EC2 instance to
create.</param>
/// <param name="keyName">The name of the key pair to associate with the
/// instance.</param>
/// <param name="groupId">The Id of the Amazon EC2 security group that will
be
/// allowed to interact with the new EC2 instance.</param>
/// <returns>The instance Id of the new EC2 instance.</returns>
public async Task<string> RunInstances(string imageId, string instanceType,
string keyName, string groupId)
{
    var request = new RunInstancesRequest
    {
        ImageId = imageId,
        InstanceType = instanceType,
        KeyName = keyName,
        MinCount = 1,
        MaxCount = 1,
        SecurityGroupIds = new List<string> { groupId }
    };
};
```

```

    var response = await _amazonEC2.RunInstancesAsync(request);
    return response.Reservation.Instances[0].InstanceId;
}

```

- Pour plus de détails sur l'API, reportez-vous [RunInstances](#) à la section Référence des AWS SDK for .NET API.

Bash

AWS CLI avec le script Bash

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

#####
# 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
}

```

Fonctions utilitaires utilisées dans cet exemple.

```

#####
# 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
}

```

- Pour plus de détails sur l'API, reportez-vous [RunInstances](#) à la section Référence des AWS CLI commandes.

C++

SDK pour C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
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();
```

- Pour plus de détails sur l'API, reportez-vous [RunInstances](#) à la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Exemple 1 : lancement d'une instance dans un sous-réseau par défaut

L'exemple `run-instances` suivant lance une instance unique de type `t2.micro` dans le sous-réseau par défaut de la région actuelle et l'associe au sous-réseau par défaut du VPC par défaut de la région. La paire de clés est facultative si vous ne prévoyez pas de vous connecter à votre instance via SSH (Linux) ou RDP (Windows).

```
aws ec2 run-instances \  
  --image-id ami-0abcdef1234567890 \  
  --instance-type t2.micro \  
  --key-name MyKeyPair
```

Sortie :

```
{  
  "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"
}
```

Exemple 2 : lancement d'une instance dans un sous-réseau autre que celui par défaut et ajout d'une adresse IP publique

L'exemple `run-instances` suivant demande une adresse IP publique pour une instance que vous lancez dans un sous-réseau autre que celui par défaut. L'instance est associée au groupe de sécurité spécifié.

```
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
```

Pour obtenir un exemple de la manière de procéder pour `run-instances`, reportez-vous à l'exemple 1.

Exemple 3 : lancement d'une instance avec des volumes supplémentaires

L'exemple `run-instances` suivant utilise un mappage de périphérique de stockage en mode bloc, spécifié dans le fichier `mapping.json`, pour attacher des volumes supplémentaires au lancement. Un mappage de périphérique de stockage en mode bloc peut spécifier des volumes EBS, des volumes de stockage d'instance ou à la fois des volumes EBS et des volumes de stockage d'instance.

```
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
```

Contenu de `mapping.json`. Cet exemple ajoute `/dev/sdh`, un volume EBS vide d'une taille de 100 Gio.

```
[  
  {  
    "DeviceName": "/dev/sdh",  
    "Ebs": {  
      "VolumeSize": 100  
    }  
  }  
]
```

```
]
```

Contenu de `mapping.json`. Cet exemple ajoute `ephemeral1` en tant que volume de stockage d'instances.

```
[
  {
    "DeviceName": "/dev/sdc",
    "VirtualName": "ephemeral1"
  }
]
```

Pour obtenir un exemple de la manière de procéder pour `run-instances`, reportez-vous à l'exemple 1.

Pour plus d'informations sur les mappages de périphérique de stockage en mode bloc, veuillez consulter la rubrique [Mappage de périphérique de stockage en mode bloc](#) dans le Guide de l'utilisateur Amazon EC2.

Exemple 4 : lancement d'une instance et ajout de balises lors de sa création

L'exemple `run-instances` suivant ajoute une balise avec une clé `webserver` et une valeur `production` à l'instance. La commande applique également une balise avec une clé de `cost-center` et une valeur de `cc123` à n'importe quel volume EBS qui est créé (dans ce cas, le volume racine).

```
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}]'
```

Pour obtenir un exemple de la manière de procéder pour `run-instances`, reportez-vous à l'exemple 1.

Exemple 5 : lancement d'une instance avec des données utilisateur

L'exemple `run-instances` suivant transmet les données de l'utilisateur dans un fichier appelé `my_script.txt` qui contient un script de configuration pour votre instance. Le script s'exécute au lancement.

```
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
```

Pour obtenir un exemple de la manière de procéder pour `run-instances`, reportez-vous à l'exemple 1.

Pour plus d'informations à propos des données utilisateur d'instance, veuillez consulter la rubrique [Utiliser les données utilisateur d'instance](#) dans le Guide de l'utilisateur Amazon EC2.

Exemple 6 : lancement d'une instance à performances extensibles

L'exemple `run-instances` suivant lance une instance `t2.micro` avec l'option de crédit `unlimited`. Lorsque vous lancez une instance T2, si vous ne spécifiez pas `--credit-specification`, l'option de crédit `standard` est utilisée par défaut. Lorsque vous lancez une instance T3, l'option de crédit `unlimited` est utilisée par défaut.

```
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
```

Pour obtenir un exemple de la manière de procéder pour `run-instances`, reportez-vous à l'exemple 1.

Pour plus d'informations à propos des instances à performances extensibles, veuillez consulter la rubrique [Instances à performances extensibles](#) dans le Guide de l'utilisateur Amazon EC2.

- Pour plus de détails sur l'API, reportez-vous [RunInstances](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.InstanceType;
import software.amazon.awssdk.services.ec2.model.RunInstancesRequest;
import software.amazon.awssdk.services.ec2.model.RunInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Tag;
import software.amazon.awssdk.services.ec2.model.CreateTagsRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 *
 * This code example requires an AMI value. You can learn more about this value
 * by reading this documentation topic:
 *
 * https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/AMIs.html
 */
public class CreateInstance {
    public static void main(String[] args) {
        final String usage = ""
```

Usage:

```
        <name> <amiId>

        Where:
            name - An instance name value that you can obtain from the AWS
Console (for example, ami-xxxxxx5c8b987b1a0).\s
            amiId - An Amazon Machine Image (AMI) value that you can
obtain from the AWS Console (for example, i-xxxxxx2734106d0ab).\s
            """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String name = args[0];
    String amiId = args[1];
    Region region = Region.US_EAST_1;
    Ec2Client ec2 = Ec2Client.builder()
        .region(region)
        .build();

    String instanceId = createEC2Instance(ec2, name, amiId);
    System.out.println("The Amazon EC2 Instance ID is " + instanceId);
    ec2.close();
}

public static String createEC2Instance(Ec2Client ec2, String name, String
amiId) {
    RunInstancesRequest runRequest = RunInstancesRequest.builder()
        .imageId(amiId)
        .instanceType(InstanceType.T1_MICRO)
        .maxCount(1)
        .minCount(1)
        .build();

    // Use a waiter to wait until the instance is running.
    System.out.println("Going to start an EC2 instance using a waiter");
    RunInstancesResponse response = ec2.runInstances(runRequest);
    String instanceIdVal = response.instances().get(0).instanceId();
    ec2.waiter().waitUntilInstanceRunning(r -> r.instanceIds(instanceIdVal));
    Tag tag = Tag.builder()
        .key("Name")
        .value(name)
        .build();
```

```
        CreateTagsRequest tagRequest = CreateTagsRequest.builder()
            .resources(instanceIdVal)
            .tags(tag)
            .build();

        try {
            ec2.createTags(tagRequest);
            System.out.printf("Successfully started EC2 Instance %s based on AMI
%s", instanceIdVal, amiId);
            return instanceIdVal;

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

        return "";
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [RunInstances](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

// Create a new EC2 instance.
export const main = async () => {
```

```
const command = new RunInstancesCommand({
  // Your key pair name.
  KeyName: "KEY_PAIR_NAME",
  // Your security group.
  SecurityGroupIds: ["SECURITY_GROUP_ID"],
  // An x86_64 compatible image.
  ImageId: "ami-0001a0d1a04bfcc30",
  // An x86_64 compatible free-tier instance type.
  InstanceType: "t1.micro",
  // Ensure only 1 instance launches.
  MinCount: 1,
  MaxCount: 1,
});

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

- Pour plus de détails sur l'API, reportez-vous [RunInstances](#) à la section Référence des AWS SDK for JavaScript API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createEC2Instance(
  name: String,
  amiId: String,
): String? {
  val request =
```

```
RunInstancesRequest {
    imageId = amiId
    instanceType = InstanceType.T1Micro
    maxCount = 1
    minCount = 1
}

Ec2Client { region = "us-west-2" }.use { ec2 ->
    val response = ec2.runInstances(request)
    val instanceId = response.instances?.get(0)?.instanceId
    val tag =
        Tag {
            key = "Name"
            value = name
        }

    val requestTags =
        CreateTagsRequest {
            resources = listOf(instanceId.toString())
            tags = listOf(tag)
        }
    ec2.createTags(requestTags)
    println("Successfully started EC2 Instance $instanceId based on AMI
$amiId")
    return instanceId
}
}
```

- Pour plus de détails sur l'API, reportez-vous [RunInstances](#) à la section AWS SDK pour la référence de l'API Kotlin.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple lance une instance unique de l'AMI spécifiée dans EC2-Classique ou dans un VPC par défaut.

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -InstanceType
m3.medium -KeyName my-key-pair -SecurityGroup my-security-group
```

Exemple 2 : Cet exemple lance une instance unique de l'AMI spécifiée dans un VPC.

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -SubnetId
  subnet-12345678 -InstanceType t2.micro -KeyName my-key-pair -SecurityGroupId
  sg-12345678
```

Exemple 3 : pour ajouter un volume EBS ou un volume de stockage d'instance, définissez un mappage de périphériques en mode bloc et ajoutez-le à la commande. Cet exemple ajoute un volume de stockage d'instance.

```
$bdm = New-Object Amazon.EC2.Model.BlockDeviceMapping
$bdm.VirtualName = "ephemeral0"
$bdm.DeviceName = "/dev/sdf"

New-EC2Instance -ImageId ami-12345678 -BlockDeviceMapping $bdm ...
```

Exemple 4 : Pour spécifier l'une des AMI Windows actuelles, obtenez son ID d'AMI à l'aide de `Get-EC2ImageByName`. Cet exemple lance une instance à partir de l'AMI de base actuelle pour Windows Server 2016.

```
$ami = Get-EC2ImageByName WINDOWS_2016_BASE

New-EC2Instance -ImageId $ami.ImageId ...
```

Exemple 5 : Lance une instance dans l'environnement hôte dédié spécifié.

```
New-EC2Instance -ImageId ami-1a2b3c4d -InstanceType m4.large -KeyName my-key-pair
  -SecurityGroupId sg-1a2b3c4d -AvailabilityZone us-west-1a -Tenancy host -HostID
  h-1a2b3c4d5e6f1a2b3
```

Exemple 6 : Cette requête lance deux instances et applique une balise avec une clé de serveur Web et une valeur de production aux instances. La demande applique également une balise avec une clé de centre de coûts et une valeur de cc123 aux volumes créés (dans ce cas, le volume racine de chaque instance).

```
$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
```

- Pour plus de détails sur l'API, reportez-vous [RunInstances](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Python

SDK pour Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance
    actions."""

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

    @classmethod
```

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

def create(self, image, instance_type, key_pair, security_groups=None):
    """
    Creates a new EC2 instance. The instance starts immediately after
    it is created.

    The instance is created in the default VPC of the current account.

    :param image: A Boto3 Image object that represents an Amazon Machine
    Image (AMI)
        that defines attributes of the instance that is created.
    The AMI
        defines things like the kind of operating system and the
    type of
        storage used by the instance.
    :param instance_type: The type of instance to create, such as 't2.micro'.
        The instance type defines things like the number of
    CPUs and
        the amount of memory.
    :param key_pair: A Boto3 KeyPair or KeyPairInfo object that represents
    the key
        pair that is used to secure connections to the instance.
    :param security_groups: A list of Boto3 SecurityGroup objects that
    represents the
        security groups that are used to grant access to
    the
        instance. When no security groups are specified,
    the
        default security group of the VPC is used.
    :return: A Boto3 Instance object that represents the newly created
    instance.
    """
    try:
        instance_params = {
            "ImageId": image.id,
            "InstanceType": instance_type,
            "KeyName": key_pair.name,
        }
        if security_groups is not None:
```

```

        instance_params["SecurityGroupIds"] = [sg.id for sg in
security_groups]
        self.instance = self.ec2_resource.create_instances(
            **instance_params, MinCount=1, MaxCount=1
        )[0]
        self.instance.wait_until_running()
    except ClientError as err:
        logging.error(
            "Couldn't create instance with image %s, instance type %s, and
key %s. "
            "Here's why: %s: %s",
            image.id,
            instance_type,
            key_pair.name,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
    else:
        return self.instance

```

- Pour plus de détails sur l'API, consultez [RunInstances](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

" Create tags for resource created during instance launch. "
DATA lt_tag specifications TYPE /aws1/
cl_ec2tag specification=>tt_tag specification list.
DATA ls_tag specifications LIKE LINE OF lt_tag specifications.

```

```

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 = 't2.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.

```

- Pour plus de détails sur l'API, reportez-vous [RunInstances](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **RunScheduledInstances** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `RunScheduledInstances`.

CLI

AWS CLI

Pour lancer une instance planifiée

Cet exemple lance l'instance planifiée spécifiée dans un VPC.

Commande :

```
aws ec2 run-scheduled-instances --scheduled-instance-id
sci-1234-1234-1234-1234-123456789012 --instance-count 1 --launch-specification
file://launch-specification.json
```

Spécification.json de lancement :

```
{
  "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"
  }
}
```

Sortie :

```
{
  "InstanceIdSet": [
    "i-1234567890abcdef0"
  ]
}
```

Cet exemple lance l'instance planifiée spécifiée dans EC2-Classic.

Commande :

```
aws ec2 run-scheduled-instances --scheduled-instance-id
sci-1234-1234-1234-1234-123456789012 --instance-count 1 --launch-specification
file://launch-specification.json
```

Spécification.json de lancement :

```
{
  "ImageId": "ami-12345678",
  "KeyName": "my-key-pair",
  "SecurityGroupIds": ["sg-12345678"],
  "InstanceType": "c4.large",
  "Placement": {
    "AvailabilityZone": "us-west-2b"
  }
  "IamInstanceProfile": {
    "Name": "my-iam-role"
  }
}
```

Sortie :

```
{
  "InstanceIdSet": [
    "i-1234567890abcdef0"
  ]
}
```

- Pour plus de détails sur l'API, reportez-vous [RunScheduledInstances](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple lance l'instance planifiée spécifiée.

```
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
```

- Pour plus de détails sur l'API, reportez-vous [RunScheduledInstances](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **StartInstances** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `StartInstances`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Démarrer avec les instances](#)

.NET

AWS SDK for .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Start an EC2 instance.
/// </summary>
/// <param name="ec2InstanceId">The instance Id of the Amazon EC2 instance
/// to start.</param>
/// <returns>Async task.</returns>
public async Task StartInstances(string ec2InstanceId)
{
```

```
var request = new StartInstancesRequest
{
    InstanceIds = new List<string> { ec2InstanceId },
};

var response = await _amazonEC2.StartInstancesAsync(request);

if (response.StartingInstances.Count > 0)
{
    var instances = response.StartingInstances;
    instances.ForEach(i =>
    {
        Console.WriteLine($"Successfully started the EC2 instance with
instance ID: {i.InstanceId}.");
    });
}
```

- Pour plus de détails sur l'API, reportez-vous [StartInstances](#) à la section Référence des AWS SDK for .NET API.

Bash

AWS CLI avec le script Bash

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
#####
# function ec2_start_instances
#
# This function starts one or more Amazon Elastic Compute Cloud (Amazon EC2)
instances.
#
# Parameters:
#     -i instance_id - The ID(s) of the instance(s) to start (comma-separated).
#     -h - Display help.
```

```

#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function ec2_start_instances() {
    local instance_ids
    local option OPTARG # Required to use getopt command in a function.

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

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

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

```

```

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

return 0
}

```

Fonctions utilitaires utilisées dans cet exemple.

```

#####
# 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
}
```

- Pour plus de détails sur l'API, reportez-vous [StartInstances](#) à la section Référence des AWS CLI commandes.

C++

SDK pour C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

Aws::EC2::Model::StartInstancesRequest start_request;
start_request.AddInstanceIds(instanceId);
start_request.SetDryRun(true);

auto dry_run_outcome = ec2Client.StartInstances(start_request);
if (dry_run_outcome.IsSuccess()) {
    std::cerr
        << "Failed dry run to start instance. A dry run should trigger an
error."
        << std::endl;
    return false;
}
```

```
else if (dry_run_outcome.GetError().GetErrorType() !=
        Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
    std::cout << "Failed dry run to start instance " << instanceId << ": "
              << dry_run_outcome.GetError().GetMessage() << std::endl;
    return false;
}

start_request.SetDryRun(false);
auto start_instancesOutcome = ec2Client.StartInstances(start_request);

if (!start_instancesOutcome.IsSuccess()) {
    std::cout << "Failed to start instance " << instanceId << ": " <<
              << start_instancesOutcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Successfully started instance " << instanceId <<
              << std::endl;
}
}
```

- Pour plus de détails sur l'API, reportez-vous [StartInstances](#) à la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Démarrage d'une instance Amazon EC2

L'exemple suivant affiche une instance basée sur Amazon EBS.

Commande :

```
aws ec2 start-instances --instance-ids i-1234567890abcdef0
```

Sortie :

```
{
  "StartingInstances": [
    {
      "InstanceId": "i-1234567890abcdef0",
```

```
        "CurrentState": {
            "Code": 0,
            "Name": "pending"
        },
        "PreviousState": {
            "Code": 80,
            "Name": "stopped"
        }
    }
}
]
```

Pour plus d'informations, veuillez consulter la rubrique Arrêt et démarrage de votre instance dans le Guide de l'utilisateur Amazon Elastic Compute Cloud.

- Pour plus de détails sur l'API, reportez-vous [StartInstances](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void startInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();

    StartInstancesRequest request = StartInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to run.
    This will take a few minutes.");
    ec2.startInstances(request);
}
```

```
DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
    .instanceIds(instanceId)
    .build();

WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceRunning(instanceRequest);
waiterResponse.matched().response().ifPresent(System.out::println);
System.out.println("Successfully started instance " + instanceId);
}
```

- Pour plus de détails sur l'API, reportez-vous [StartInstances](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

export const main = async () => {
    const command = new StartInstancesCommand({
        // Use DescribeInstancesCommand to find InstanceIds
        InstanceIds: ["INSTANCE_ID"],
    });

    try {
        const { StartingInstances } = await client.send(command);
        const instanceIdList = StartingInstances.map(
            (instance) => ` • ${instance.InstanceId}`,
        );
        console.log("Starting instances:");
    }
}
```

```
console.log(instanceIdList.join("\n"));
} catch (err) {
  console.error(err);
}
};
```

- Pour plus de détails sur l'API, reportez-vous [StartInstances](#) à la section Référence des AWS SDK for JavaScript API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun startInstanceSc(instanceId: String) {
    val request =
        StartInstancesRequest {
            instanceIds = listOf(instanceId)
        }

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

- Pour plus de détails sur l'API, reportez-vous [StartInstances](#) à la section AWS SDK pour la référence de l'API Kotlin.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple démarre l'instance spécifiée.

```
Start-EC2Instance -InstanceId i-12345678
```

Sortie :

CurrentState	InstanceId	PreviousState
-----	-----	-----
Amazon.EC2.Model.InstanceState	i-12345678	Amazon.EC2.Model.InstanceState

Exemple 2 : Cet exemple démarre les instances spécifiées.

```
@("i-12345678", "i-76543210") | Start-EC2Instance
```

Exemple 3 : Cet exemple démarre l'ensemble des instances actuellement arrêtées. Les objets d'instance renvoyés par Get-EC2Instance sont redirigés vers Start-EC2Instance. La syntaxe utilisée dans cet exemple nécessite PowerShell la version 3 ou supérieure.

```
(Get-EC2Instance -Filter @{ Name="instance-state-name";  
Values="stopped"}).Instances | Start-EC2Instance
```

Exemple 4 : avec PowerShell la version 2, vous devez utiliser New-Object pour créer le filtre pour le paramètre Filter.

```
$filter = New-Object Amazon.EC2.Model.Filter  
$filter.Name = "instance-state-name"  
$filter.Values = "stopped"  
  
(Get-EC2Instance -Filter $filter).Instances | Start-EC2Instance
```

- Pour plus de détails sur l'API, reportez-vous [StartInstances](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Python

SDK pour Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance
    actions."""

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

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

    def start(self):
        """
        Starts an instance and waits for it to be in a running state.

        :return: The response to the start request.
        """
        if self.instance is None:
            logger.info("No instance to start.")
            return
```

```
try:
    response = self.instance.start()
    self.instance.wait_until_running()
except ClientError as err:
    logger.error(
        "Couldn't start instance %s. Here's why: %s: %s",
        self.instance.id,
        err.response["Error"]["Code"],
        err.response["Error"]["Message"],
    )
    raise
else:
    return response
```

- Pour plus de détails sur l'API, consultez [StartInstances](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
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."
return true
rescue StandardError => e
  puts "Error starting instance: #{e.message}"
  return false
end

# Example usage:
def run_me
  instance_id = ""
  region = ""
  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
    puts "Usage:  ruby ec2-ruby-example-start-instance-i-123abc.rb " \
      "INSTANCE_ID REGION "
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  puts "Example: ruby ec2-ruby-example-start-instance-i-123abc.rb " \
```

```
    "i-123abc us-west-2"
  exit 1
# If no values are specified at the command prompt, use these default values.
# Replace us-west-2 with the AWS Region you're using for Amazon EC2.
elsif ARGV.count.zero?
  instance_id = "i-123abc"
  region = "us-west-2"
# Otherwise, use the values as specified at the command prompt.
else
  instance_id = ARGV[0]
  region = ARGV[1]
end

ec2_client = Aws::EC2::Client.new(region: region)

puts "Attempting to start instance '#{instance_id}' " \
      "(this might take a few minutes)..."
unless instance_started?(ec2_client, instance_id)
  puts "Could not start instance."
end
end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Pour plus de détails sur l'API, reportez-vous [StartInstances](#) à la section Référence des AWS SDK for Ruby API.

Rust

SDK pour Rust

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
async fn start_instance(client: &Client, id: &str) -> Result<(), Error> {
  // start_instance has no unique errors to handle.
  client.start_instances().instance_ids(id).send().await?;
```

```
println!("Waiting for instance to be running");

let wait_for_running = client
    .wait_until_instance_running()
    .instance_ids(id)
    .wait(Duration::from_secs(60))
    .await;

match wait_for_running {
    Ok(_) => println!("Instance is running"),
    Err(err) => match err {
        WaiterError::ExceededMaxWait(exceeded) => {
            println!(
                "Exceeded max time waiting for instance to start. Exceeded
{}s by {}s.",
                exceeded.max_wait().as_secs(),
                (exceeded.elapsed() - exceeded.max_wait()).as_secs()
            );
            return Ok(());
        }
        _ => return Err(err.into()),
    },
}

println!("Started instance.");

Ok(())
}
```

- Pour plus de détails sur l'API, voir [StartInstances](#) la section de référence de l'API AWS SDK pour Rust.

SAP ABAP

Kit SDK pour SAP ABAP

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
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.
```

- Pour plus de détails sur l'API, reportez-vous [StartInstances](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **StopInstances** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `StopInstances`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Démarrer avec les instances](#)

.NET

AWS SDK for .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

```
public async Task StopInstances(string ec2InstanceId)
{
    // In addition to the list of instance Ids, the
    // request can also include the following properties:
    //     Force      When true, forces the instances to
    //                 stop but you must check the integrity
    //                 of the file system. Not recommended on
    //                 Windows instances.
    //     Hibernate  When true, hibernates the instance if the
    //                 instance was enabled for hibernation when
    //                 it was launched.
    var request = new StopInstancesRequest
    {
        InstanceIds = new List<string> { ec2InstanceId },
    };

    var response = await _amazonEC2.StopInstancesAsync(request);

    if (response.StoppingInstances.Count > 0)
    {
        var instances = response.StoppingInstances;
        instances.ForEach(i =>
        {
            Console.WriteLine($"Successfully stopped the EC2 Instance " +
                $"with InstanceID: {i.InstanceId}.");
        });
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [StopInstances](#) à la section Référence des AWS SDK for .NET API.

Bash

AWS CLI avec le script Bash

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

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

```

export OPTIND=1

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

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

return 0
}

```

Fonctions utilitaires utilisées dans cet exemple.

```

#####
# 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
}
```

- Pour plus de détails sur l'API, reportez-vous [StopInstances](#) à la section Référence des AWS CLI commandes.

C++

SDK pour C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::StopInstancesRequest request;
request.AddInstanceIds(instanceId);
request.SetDryRun(true);
```

```
    auto dry_run_outcome = ec2Client.StopInstances(request);
    if (dry_run_outcome.IsSuccess()) {
        std::cerr
            << "Failed dry run to stop instance. A dry run should trigger an
error."
            << std::endl;
        return false;
    }
    else if (dry_run_outcome.GetError().GetErrorType() !=
        Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to stop instance " << instanceId << ": "
            << dry_run_outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    request.SetDryRun(false);
    auto outcome = ec2Client.StopInstances(request);
    if (!outcome.IsSuccess()) {
        std::cout << "Failed to stop instance " << instanceId << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully stopped instance " << instanceId <<
            std::endl;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [StopInstances](#) à la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Exemple 1 : arrêt d'une instance Amazon EC2

L'exemple `stop-instances` suivant arrête l'instance Amazon EBS spécifiée.

```
aws ec2 stop-instances \
    --instance-ids i-1234567890abcdef0
```

Sortie :

```
{
  "StoppingInstances": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "CurrentState": {
        "Code": 64,
        "Name": "stopping"
      },
      "PreviousState": {
        "Code": 16,
        "Name": "running"
      }
    }
  ]
}
```

Pour plus d'informations, veuillez consulter la rubrique [Arrêt et démarrage de votre instance](#) dans le Guide de l'utilisateur Amazon Elastic Compute Cloud.

Exemple 2 : mise en veille prolongée d'une instance Amazon EC2

L'exemple `stop-instances` suivant met en hibernation l'instance soutenue par Amazon EBS si l'instance est activée pour l'hibernation et remplit les conditions préalables à l'hibernation. L'instance est arrêtée après avoir été mise en hibernation.

```
aws ec2 stop-instances \
  --instance-ids i-1234567890abcdef0 \
  --hibernate
```

Sortie :

```
{
  "StoppingInstances": [
    {
      "CurrentState": {
        "Code": 64,
        "Name": "stopping"
      },
      "InstanceId": "i-1234567890abcdef0",
      "PreviousState": {
        "Code": 16,
        "Name": "running"
      }
    }
  ]
}
```

```

    }
  }
]
}

```

Pour plus d'informations, veuillez consulter la rubrique [Mise en veille prolongée de votre instance à la demande Linux](#) dans le Guide de l'utilisateur Amazon Elastic Compute Cloud.

- Pour plus de détails sur l'API, reportez-vous [StopInstances](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

public static void stopInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();
    StopInstancesRequest request = StopInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to stop.
This will take a few minutes.");
    ec2.stopInstances(request);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceStopped(instanceRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Successfully stopped instance " + instanceId);
}

```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [StopInstances](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

export const main = async () => {
  const command = new StopInstancesCommand({
    // Use DescribeInstancesCommand to find InstanceIds
    InstanceIds: ["INSTANCE_ID"],
  });

  try {
    const { StoppingInstances } = await client.send(command);
    const instanceIdList = StoppingInstances.map(
      (instance) => ` • ${instance.InstanceId}`,
    );
    console.log("Stopping instances:");
    console.log(instanceIdList.join("\n"));
  } catch (err) {
    console.error(err);
  }
};
```

- Pour plus de détails sur l'API, reportez-vous [StopInstances](#) à la section Référence des AWS SDK for JavaScript API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

- Pour plus de détails sur l'API, reportez-vous [StopInstances](#) à la section AWS SDK pour la référence de l'API Kotlin.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple arrête l'instance spécifiée.

```
Stop-EC2Instance -InstanceId i-12345678
```

Sortie :

CurrentState	InstanceId	PreviousState
-----	-----	-----
Amazon.EC2.Model.InstanceState	i-12345678	Amazon.EC2.Model.InstanceState

- Pour plus de détails sur l'API, reportez-vous [StopInstances](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Python

SDK pour Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance
    actions."""

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

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

    def stop(self):
```

```
"""
Stops an instance and waits for it to be in a stopped state.

:return: The response to the stop request.
"""
if self.instance is None:
    logger.info("No instance to stop.")
    return

try:
    response = self.instance.stop()
    self.instance.wait_until_stopped()
except ClientError as err:
    logger.error(
        "Couldn't stop instance %s. Here's why: %s: %s",
        self.instance.id,
        err.response["Error"]["Code"],
        err.response["Error"]["Message"],
    )
    raise
else:
    return response
```

- Pour plus de détails sur l'API, consultez [StopInstances](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
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."
return true
rescue StandardError => e
  puts "Error stopping instance: #{e.message}"
  return false
end

# Example usage:
def run_me
  instance_id = ""
  region = ""
```

```
# Print usage information and then stop.
if ARGV[0] == "--help" || ARGV[0] == "-h"
  puts "Usage:  ruby ec2-ruby-example-stop-instance-i-123abc.rb " \
    "INSTANCE_ID REGION "
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  puts "Example: ruby ec2-ruby-example-start-instance-i-123abc.rb " \
    "i-123abc us-west-2"
  exit 1
# If no values are specified at the command prompt, use these default values.
# Replace us-west-2 with the AWS Region you're using for Amazon EC2.
elsif ARGV.count.zero?
  instance_id = "i-123abc"
  region = "us-west-2"
# Otherwise, use the values as specified at the command prompt.
else
  instance_id = ARGV[0]
  region = ARGV[1]
end

ec2_client = Aws::EC2::Client.new(region: region)

puts "Attempting to stop instance '#{instance_id}' " \
  "(this might take a few minutes)..."
unless instance_stopped?(ec2_client, instance_id)
  puts "Could not stop instance."
end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Pour plus de détails sur l'API, reportez-vous [StopInstances](#) à la section Référence des AWS SDK for Ruby API.

Rust

SDK pour Rust

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
async fn stop_instance(client: &Client, id: &str) -> Result<(), Error> {
    client.stop_instances().instance_ids(id).send().await?;

    println!("Stopping instance...");

    let wait = client
        .wait_until_instance_stopped()
        .instance_ids(id)
        .wait(Duration::from_secs(60))
        .await;

    match wait {
        Ok(_) => {
            println!("Stopped instance.");
        }
        Err(err) => match err {
            WaiterError::ExceededMaxWait(exceeded) => {
                println!(
                    "Exceeded max time waiting for instance to stop. Exceeded {}s
by {}s",
                    exceeded.max_wait().as_secs(),
                    (exceeded.elapsed() - exceeded.max_wait()).as_secs()
                );
            }
            _ => return Err(err.into()),
        },
    };
    Ok(())
}
```

- Pour plus de détails sur l'API, voir [StopInstances](#) la section de référence de l'API AWS SDK for Rust.

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
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.
```

- Pour plus de détails sur l'API, reportez-vous [StopInstances](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **TerminateInstances** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `TerminateInstances`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Démarrer avec les instances](#)

.NET

AWS SDK for .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Terminate an EC2 instance.
```

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

    var response = await _amazonEC2.TerminateInstancesAsync(request);
    return response.TerminatingInstances;
}
```

- Pour plus de détails sur l'API, reportez-vous [TerminateInstances](#) à la section Référence des AWS SDK for .NET API.

Bash

AWS CLI avec le script Bash

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
#####
# 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
}

```

Fonctions utilitaires utilisées dans cet exemple.

```

#####
# 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."
    fi
}

```

```
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
}
```

- Pour plus de détails sur l'API, reportez-vous [TerminateInstances](#) à la section Référence des AWS CLI commandes.

C++

SDK pour C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
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;
}
```

- Pour plus de détails sur l'API, reportez-vous [TerminateInstances](#) à la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Résiliation d'une instance Amazon EC2

Cet exemple résilie l'instance spécifiée.

Commande :

```
aws ec2 terminate-instances --instance-ids i-1234567890abcdef0
```

Sortie :

```
{
  "TerminatingInstances": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "CurrentState": {
        "Code": 32,
        "Name": "shutting-down"
      },
      "PreviousState": {
        "Code": 16,
        "Name": "running"
      }
    }
  ]
}
```

Pour plus d'informations, veuillez consulter la rubrique Using Amazon EC2 Instances dans le Guide de l'utilisateur AWS Command Line Interface.

- Pour plus de détails sur l'API, reportez-vous [TerminateInstances](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void terminateEC2(Ec2Client ec2, String instanceId) {
    try {
        Ec2Waiter ec2Waiter = Ec2Waiter.builder()
            .overrideConfiguration(b -> b.maxAttempts(100))
            .client(ec2)
            .build();

        TerminateInstancesRequest ti = TerminateInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        System.out.println("Use an Ec2Waiter to wait for the instance to
terminate. This will take a few minutes.");
        ec2.terminateInstances(ti);
        DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        WaiterResponse<DescribeInstancesResponse> waiterResponse = ec2Waiter
            .waitUntilInstanceTerminated(instanceRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully started instance " + instanceId);
        System.out.println(instanceId + " is terminated!");
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }  
  }  
}
```

- Pour plus de détails sur l'API, reportez-vous [TerminateInstances](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import { TerminateInstancesCommand } from "@aws-sdk/client-ec2";  
  
import { client } from "../libs/client.js";  
  
export const main = async () => {  
  const command = new TerminateInstancesCommand({  
    InstanceIds: ["INSTANCE_ID"],  
  });  
  
  try {  
    const { TerminatingInstances } = await client.send(command);  
    const instanceList = TerminatingInstances.map(  
      (instance) => ` • ${instance.InstanceId}`,  
    );  
    console.log("Terminating instances:");  
    console.log(instanceList.join("\n"));  
  } catch (err) {  
    console.error(err);  
  }  
};
```

- Pour plus de détails sur l'API, reportez-vous [TerminateInstances](#) à la section Référence des AWS SDK for JavaScript API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun terminateEC2(instanceID: String) {
    val request =
        TerminateInstancesRequest {
            instanceIds = listOf(instanceID)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.terminateInstances(request)
        response.terminatingInstances?.forEach { instance ->
            println("The ID of the terminated instance is
                ${instance.instanceId}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [TerminateInstances](#) à la section AWS SDK pour la référence de l'API Kotlin.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple met fin à l'instance spécifiée (l'instance est peut-être en cours d'exécution ou à l'état « arrêté »). L'applet de commande demandera une confirmation avant de continuer ; utilisez le commutateur `-Force` pour supprimer l'invite.

```
Remove-EC2Instance -InstanceId i-12345678
```

Sortie :

CurrentState	InstanceId	PreviousState
-----	-----	-----
Amazon.EC2.Model.InstanceState	i-12345678	Amazon.EC2.Model.InstanceState

- Pour plus de détails sur l'API, reportez-vous [TerminateInstances](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Python

SDK pour Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance
    actions."""

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

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

    def terminate(self):
```

```
"""
Terminates an instance and waits for it to be in a terminated state.
"""
if self.instance is None:
    logger.info("No instance to terminate.")
    return

instance_id = self.instance.id
try:
    self.instance.terminate()
    self.instance.wait_until_terminated()
    self.instance = None
except ClientError as err:
    logging.error(
        "Couldn't terminate instance %s. Here's why: %s: %s",
        instance_id,
        err.response["Error"]["Code"],
        err.response["Error"]["Message"],
    )
    raise
```

- Pour plus de détails sur l'API, consultez [TerminateInstances](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
require "aws-sdk-ec2"

# Prerequisites:
#
```

```
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was terminated; otherwise, false.
# @example
#   exit 1 unless instance_terminated?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'i-123abc'
#   )
def instance_terminated?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

  if response.instance_statuses.count.positive? &&
    response.instance_statuses[0].instance_state.name == "terminated"

    puts "The instance is already terminated."
    return true
  end

  ec2_client.terminate_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_terminated, instance_ids: [instance_id])
  puts "Instance terminated."
  return true
rescue StandardError => e
  puts "Error terminating instance: #{e.message}"
  return false
end

# Example usage:
def run_me
  instance_id = ""
  region = ""
  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
    puts "Usage:  ruby ec2-ruby-example-terminate-instance-i-123abc.rb " \
      "INSTANCE_ID REGION "
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts "Example: ruby ec2-ruby-example-terminate-instance-i-123abc.rb " \
      "i-123abc us-west-2"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
```

```
instance_id = "i-123abc"
region = "us-west-2"
# Otherwise, use the values as specified at the command prompt.
else
  instance_id = ARGV[0]
  region = ARGV[1]
end

ec2_client = Aws::EC2::Client.new(region: region)

puts "Attempting to terminate instance '#{instance_id}' " \
      "(this might take a few minutes)..."
unless instance_terminated?(ec2_client, instance_id)
  puts "Could not terminate instance."
end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Pour plus de détails sur l'API, reportez-vous [TerminateInstances](#) à la section Référence des AWS SDK for Ruby API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **UnassignPrivateIpAddresses** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `UnassignPrivateIpAddresses`.

CLI

AWS CLI

Pour annuler l'attribution d'une adresse IP privée secondaire depuis une interface réseau

Cet exemple annule l'attribution de l'adresse IP privée spécifiée à l'interface réseau spécifiée. Si la commande réussit, aucune sortie n'est renvoyée.

Commande :

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

- Pour plus de détails sur l'API, reportez-vous [UnassignPrivateIpAddresses](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell

Exemple 1 : Cet exemple annule l'attribution de l'adresse IP privée spécifiée à l'interface réseau spécifiée.

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

- Pour plus de détails sur l'API, reportez-vous [UnassignPrivateIpAddresses](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **UnmonitorInstances** avec un AWS SDK ou une CLI

Les exemples de code suivants montrent comment utiliser `UnmonitorInstances`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Démarrer avec les instances](#)

C++

SDK pour C++

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::UnmonitorInstancesRequest unrequest;
unrequest.AddInstanceIds(instanceId);
unrequest.SetDryRun(true);

auto undryRunOutcome = ec2Client.UnmonitorInstances(unrequest);
if (undryRunOutcome.IsSuccess()) {
    std::cerr
        << "Failed dry run to disable monitoring on instance. A dry run
should trigger an error."
        <<
        std::endl;
    return false;
}
else if (undryRunOutcome.GetError().GetErrorType() !=
    Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
    std::cout << "Failed dry run to disable monitoring on instance " <<
        instanceId << ": " << undryRunOutcome.GetError().GetMessage()
<<
        std::endl;
    return false;
}

unrequest.SetDryRun(false);
auto unmonitorInstancesOutcome = ec2Client.UnmonitorInstances(unrequest);
if (!unmonitorInstancesOutcome.IsSuccess()) {
    std::cout << "Failed to disable monitoring on instance " << instanceId
        << ": " << unmonitorInstancesOutcome.GetError().GetMessage() <<
        std::endl;
}
else {
    std::cout << "Successfully disable monitoring on instance " <<
```

```
        instanceId << std::endl;
    }
```

- Pour plus de détails sur l'API, reportez-vous [UnmonitorInstances](#) à la section Référence des AWS SDK for C++ API.

CLI

AWS CLI

Désactivation de la surveillance détaillée d'une instance

Cet exemple de commande désactive la surveillance détaillée de l'instance spécifiée.

Commande :

```
aws ec2 unmonitor-instances --instance-ids i-1234567890abcdef0
```

Sortie :

```
{
  "InstanceMonitorings": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "Monitoring": {
        "State": "disabling"
      }
    }
  ]
}
```

- Pour plus de détails sur l'API, reportez-vous [UnmonitorInstances](#) à la section Référence des AWS CLI commandes.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

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

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

export const main = async () => {
  const command = new UnmonitorInstancesCommand({
    InstanceIds: ["i-09a3dfe7ae00e853f"],
  });

  try {
    const { InstanceMonitorings } = await client.send(command);
    const instanceMonitoringsList = InstanceMonitorings.map(
      (im) =>
        ` • Detailed monitoring state for ${im.InstanceId} is
        ${im.Monitoring.State}.`,
    );
    console.log("Monitoring status:");
    console.log(instanceMonitoringsList.join("\n"));
  } catch (err) {
    console.error(err);
  }
};
```

- Pour plus de détails sur l'API, reportez-vous [UnmonitorInstances](#) à la section Référence des AWS SDK for JavaScript API.

PowerShell

Outils pour PowerShell

Exemple 1 : cet exemple désactive la surveillance détaillée pour l'instance spécifiée.

```
Stop-EC2InstanceMonitoring -InstanceId i-12345678
```

Sortie :

```
InstanceId      Monitoring
-----
i-12345678     Amazon.EC2.Model.Monitoring
```

- Pour plus de détails sur l'API, reportez-vous [UnmonitorInstances](#) à la section Référence des AWS Tools for PowerShell applets de commande.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Scénarios pour Amazon EC2 utilisant des kits de développement logiciel AWS

Les exemples de code suivants vous montrent comment implémenter des scénarios courants dans Amazon EC2 à l'aide AWS de kits SDK. Ces scénarios vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions dans Amazon EC2. Chaque scénario inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code.

Exemples

- [Créez et gérez un service résilient à l'aide d'un AWS SDK](#)
- [Commencez à utiliser les instances Amazon EC2 à l'aide d'un SDK AWS](#)

Créez et gérez un service résilient à l'aide d'un AWS SDK

Les exemples de code suivant montrent comment créer un service web à charge équilibrée qui renvoie des recommandations de livres, de films et de chansons. L'exemple montre comment le

service répond aux défaillances et comment le restructurer pour accroître la résilience en cas de défaillance.

- Utilisez un groupe Amazon EC2 Auto Scaling pour créer des instances Amazon Elastic Compute Cloud (Amazon EC2) sur la base d'un modèle de lancement et pour maintenir le nombre d'instances dans une plage spécifiée.
- Gérez et distribuez les requêtes HTTP avec Elastic Load Balancing.
- Surveillez l'état des instances d'un groupe Auto Scaling et transférez les demandes uniquement aux instances saines.
- Exécutez un serveur Web Python sur chaque instance EC2 pour gérer les requêtes HTTP. Le serveur Web répond par des recommandations et des surveillances de l'état.
- Simulez un service de recommandation avec une table Amazon DynamoDB.
- Contrôlez la réponse du serveur Web aux demandes et aux contrôles de santé en mettant à jour AWS Systems Manager les paramètres.

.NET

AWS SDK for .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Exécutez un scénario interactif à une invite de commande.

```
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($"\\thttp://{endPoint}\\n");
        }
        else
        {
            Console.WriteLine(
                "\\nCouldn't get a successful response from the load balancer
endpoint. Troubleshoot by\\n"
                + "manually verifying that your VPC and security group are
configured correctly and that\\n"
                + "you can successfully make a GET request to the load balancer
endpoint:\\n");
            Console.WriteLine($"\\thttp://{endPoint}\\n");
        }
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue with the
demo.");
        if (interactive)
            Console.ReadLine();
        return true;
    }

    /// <summary>
    /// Demonstrate the steps of the scenario.
    /// </summary>
    /// <param name="interactive">True to run as an interactive scenario.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> Demo(bool interactive)
    {
        var ssmOnlyPolicy = Path.Join(_configuration["resourcePath"],
            "ssm_only_policy.json");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Resetting parameters to starting values for demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine("\\nThis part of the demonstration shows how to toggle
different parts of the system\\n" +
            "to create situations where the web service fails, and
shows how using a resilient\\n" +
```

```
        "architecture can keep the web service running in spite
of these failures.");
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine($"The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.\n" +
            $"The table name is contained in a Systems Manager
parameter named '{_smParameterWrapper.TableParameter}'.\n" +
            $"To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");
        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as\n" +
            "healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Instead of failing when the recommendation service
fails, the web service can return a static response.");
        Console.WriteLine("While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.FailureResponseParameter,
"static");

        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a static response.");
        Console.WriteLine("The service still reports as healthy because health
checks are still shallow.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Let's reinstate the recommendation service.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
_smParameterWrapper.TableName);
```

```
    Console.WriteLine(
        "\nLet's also substitute bad credentials for one of the instances in
the target group so that it can't\n" +
        "access the DynamoDB recommendation table.\n"
    );
    await _autoScalerWrapper.CreateInstanceProfileWithName(
        _autoScalerWrapper.BadCredsPolicyName,
        _autoScalerWrapper.BadCredsRoleName,
        _autoScalerWrapper.BadCredsProfileName,
        ssmOnlyPolicy,
        new List<string> { "AmazonSSMManagedInstanceCore" }
    );
    var instances = await
_autoScalerWrapper.GetInstancesByGroupName(_autoScalerWrapper.GroupName);
    var badInstanceId = instances.First();
    var instanceProfile = await
_autoScalerWrapper.GetInstanceProfile(badInstanceId);
    Console.WriteLine(
        $"Replacing the profile for instance {badInstanceId} with a profile
that contains\n" +
        "bad credentials...\n"
    );
    await _autoScalerWrapper.ReplaceInstanceProfile(
        badInstanceId,
        _autoScalerWrapper.BadCredsProfileName,
        instanceProfile.AssociationId
    );
    Console.WriteLine(
        "Now, sending a GET request to the load balancer endpoint returns
either a recommendation or a static response,\n" +
        "depending on which instance is selected by the load balancer.\n"
    );
    if (interactive)
        await DemoActionChoices();

    Console.WriteLine("\nLet's implement a deep health check. For this demo,
a deep health check tests whether");
    Console.WriteLine("the web service can access the DynamoDB table that it
depends on for recommendations. Note that");
    Console.WriteLine("the deep health check is only for ELB routing and not
for Auto Scaling instance health.");
    Console.WriteLine("This kind of deep health check is not recommended for
Auto Scaling instance health, because it");
```

```
    Console.WriteLine("risks accidental termination of all instances in the
Auto Scaling group when a dependent service fails.");

    Console.WriteLine("\nBy implementing deep health checks, the load
balancer can detect when one of the instances is failing");
    Console.WriteLine("and take that instance out of rotation.");

    await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.HealthCheckParameter,
"deep");

    Console.WriteLine($"Now, checking target health indicates that the
instance with bad credentials ({badInstanceId})");
    Console.WriteLine("is unhealthy. Note that it might take a minute or two
for the load balancer to detect the unhealthy");
    Console.WriteLine("instance. Sending a GET request to the load balancer
endpoint always returns a recommendation, because");
    Console.WriteLine("the load balancer takes unhealthy instances out of its
rotation.");

    if (interactive)
        await DemoActionChoices();

    Console.WriteLine("\nBecause the instances in this demo are controlled by
an auto scaler, the simplest way to fix an unhealthy");
    Console.WriteLine("instance is to terminate it and let the auto scaler
start a new instance to replace it.");

    await _autoScalerWrapper.TryTerminateInstanceById(badInstanceId);

    Console.WriteLine($"Even while the instance is terminating and the new
instance is starting, sending a GET");
    Console.WriteLine("request to the web service continues to get a
successful recommendation response because");
    Console.WriteLine("starts and reports as healthy, it is included in the
load balancing rotation.");
    Console.WriteLine("Note that terminating and replacing an instance
typically takes several minutes, during which time you");
    Console.WriteLine("can see the changing health check status until the new
instance is running and healthy.");

    if (interactive)
        await DemoActionChoices();
```

```
        Console.WriteLine("\nIf the recommendation service fails now, deep health
checks mean all instances report as unhealthy.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");

        Console.WriteLine($" \nWhen all instances are unhealthy, the load balancer
continues to route requests even to");
        Console.WriteLine("unhealthy instances, allowing them to fail open and
return a static response rather than fail");
        Console.WriteLine("closed and report failure to the customer.");

        if (interactive)
            await DemoActionChoices();
        await _smParameterWrapper.Reset();

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to ask the user for cleanup.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> DestroyResources(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources\n" +
            "that were created for this demo."
        );

        if (!interactive || GetYesNoResponse("Do you want to clean up all demo
resources? (y/n) "))
        {
            await
_elasticLoadBalancerWrapper.DeleteLoadBalancerByName(_elasticLoadBalancerWrapper.LoadBal
            await
_elasticLoadBalancerWrapper.DeleteTargetGroupByName(_elasticLoadBalancerWrapper.TargetGr
            await
_autoScalerWrapper.TerminateAndDeleteAutoScalingGroupWithName(_autoScalerWrapper.GroupNa
```

```

        await
        _autoScalerWrapper.DeleteKeyPairByName(_autoScalerWrapper.KeyPairName);
        await
        _autoScalerWrapper.DeleteTemplateByName(_autoScalerWrapper.LaunchTemplateName);
        await _autoScalerWrapper.DeleteInstanceProfile(
            _autoScalerWrapper.BadCredsProfileName,
            _autoScalerWrapper.BadCredsRoleName
        );
        await
        _recommendations.DestroyDatabaseByName(_recommendations.TableName);
    }
    else
    {
        Console.WriteLine(
            "Ok, we'll leave the resources intact.\n" +
            "Don't forget to delete them when you're done with them or you
            might incur unexpected charges."
        );
    }

    Console.WriteLine(new string('-', 80));
    return true;
}

```

Créez une classe qui englobe les actions Auto Scaling et Amazon EC2.

```

/// <summary>
/// Encapsulates Amazon EC2 Auto Scaling and EC2 management methods.
/// </summary>
public class AutoScalerWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;
    private readonly IAmazonEC2 _amazonEc2;
    private readonly IAmazonSimpleSystemsManagement _amazonSsm;
    private readonly IAmazonIdentityManagementService _amazonIam;

    private readonly string _instanceType = "";
    private readonly string _amiParam = "";
    private readonly string _launchTemplateName = "";
    private readonly string _groupName = "";
    private readonly string _instancePolicyName = "";
    private readonly string _instanceRoleName = "";
}

```

```
private readonly string _instanceProfileName = "";
private readonly string _badCredsProfileName = "";
private readonly string _badCredsRoleName = "";
private readonly string _badCredsPolicyName = "";
private readonly string _keyPairName = "";

public string GroupName => _groupName;
public string KeyPairName => _keyPairName;
public string LaunchTemplateName => _launchTemplateName;
public string InstancePolicyName => _instancePolicyName;
public string BadCredsProfileName => _badCredsProfileName;
public string BadCredsRoleName => _badCredsRoleName;
public string BadCredsPolicyName => _badCredsPolicyName;

/// <summary>
/// Constructor for the AutoScalerWrapper.
/// </summary>
/// <param name="amazonAutoScaling">The injected AutoScaling client.</param>
/// <param name="amazonEc2">The injected EC2 client.</param>
/// <param name="amazonIam">The injected IAM client.</param>
/// <param name="amazonSsm">The injected SSM client.</param>
public AutoScalerWrapper(
    IAmazonAutoScaling amazonAutoScaling,
    IAmazonEC2 amazonEc2,
    IAmazonSimpleSystemsManagement amazonSsm,
    IAmazonIdentityManagementService amazonIam,
    IConfiguration configuration)
{
    _amazonAutoScaling = amazonAutoScaling;
    _amazonEc2 = amazonEc2;
    _amazonSsm = amazonSsm;
    _amazonIam = amazonIam;

    var prefix = configuration["resourcePrefix"];
    _instanceType = configuration["instanceType"];
    _amiParam = configuration["amiParam"];

    _launchTemplateName = prefix + "-template";
    _groupName = prefix + "-group";
    _instancePolicyName = prefix + "-pol";
    _instanceRoleName = prefix + "-role";
    _instanceProfileName = prefix + "-prof";
    _badCredsPolicyName = prefix + "-bc-pol";
    _badCredsRoleName = prefix + "-bc-role";
}
```

```

        _badCredsProfileName = prefix + "-bc-prof";
        _keyPairName = prefix + "-key-pair";
    }

    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with
    a specified name.
    /// An instance's associated profile defines a role that is assumed by the
    /// instance. The role has attached policies that specify the AWS permissions
    granted to
    /// clients that run on the instance.
    /// </summary>
    /// <param name="policyName">Name to use for the policy.</param>
    /// <param name="roleName">Name to use for the role.</param>
    /// <param name="profileName">Name to use for the profile.</param>
    /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
    /// <param name="awsManagedPolicies">AWS Managed policies to be attached to
    the role.</param>
    /// <returns>The Arn of the profile.</returns>
    public async Task<string> CreateInstanceProfileWithName(
        string policyName,
        string roleName,
        string profileName,
        string ssmOnlyPolicyFile,
        List<string>? awsManagedPolicies = null)
    {

        var assumeRoleDoc = "{" +
            "\"Version\": \"2012-10-17\"," +
            "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": [" +
                    "\"ec2.amazonaws.com\"" +
                "]" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
            "}]]" +
            "};

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";

```

```
try
{
    var createPolicyResult = await _amazonIam.CreatePolicyAsync(
        new CreatePolicyRequest
        {
            PolicyName = policyName,
            PolicyDocument = policyDocument
        });
    policyArn = createPolicyResult.Policy.Arn;
}
catch (EntityAlreadyExistsException)
{
    // The policy already exists, so we look it up to get the Arn.
    var policiesPaginator = _amazonIam.Paginators.ListPolicies(
        new ListPoliciesRequest()
        {
            Scope = PolicyScopeType.Local
        });
    // Get the entire list using the paginator.
    await foreach (var policy in policiesPaginator.Policies)
    {
        if (policy.PolicyName.Equals(policyName))
        {
            policyArn = policy.Arn;
        }
    }

    if (policyArn == null)
    {
        throw new InvalidOperationException("Policy not found");
    }
}

try
{
    await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
    {
        RoleName = roleName,
        AssumeRolePolicyDocument = assumeRoleDoc,
    });
    await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
    {
        RoleName = roleName,
        PolicyArn = policyArn
    });
}
```

```
    });
    if (awsManagedPolicies != null)
    {
        foreach (var awsPolicy in awsManagedPolicies)
        {
            await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
            {
                PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                RoleName = roleName
            });
        }
    }
}
catch (EntityAlreadyExistsException)
{
    Console.WriteLine("Role already exists.");
}

string profileArn = "";
try
{
    var profileCreateResponse = await
_amazonIam.CreateInstanceProfileAsync(
    new CreateInstanceProfileRequest()
    {
        InstanceProfileName = profileName
    });
    // Allow time for the profile to be ready.
    profileArn = profileCreateResponse.InstanceProfile.Arn;
    Thread.Sleep(10000);
    await _amazonIam.AddRoleToInstanceProfileAsync(
    new AddRoleToInstanceProfileRequest()
    {
        InstanceProfileName = profileName,
        RoleName = roleName
    });
}
catch (EntityAlreadyExistsException)
{
    Console.WriteLine("Policy already exists.");
    var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
    new GetInstanceProfileRequest()
```

```
        {
            InstanceProfileName = profileName
        });
        profileArn = profileGetResponse.InstanceProfile.Arn;
    }
    return profileArn;
}

/// <summary>
/// Create a new key pair and save the file.
/// </summary>
/// <param name="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 = newKeyName });
        await File.WriteAllTextAsync($"{newKeyName}.pem",
            keyResponse.KeyPair.KeyMaterial);
        Console.WriteLine($"Created key pair {newKeyName}.");
    }
    catch (AlreadyExistsException)
    {
        Console.WriteLine("Key pair already exists.");
    }
}

/// <summary>
/// Delete the key pair and file by name.
/// </summary>
/// <param name="deleteKeyName">The key pair to delete.</param>
/// <returns>Async task.</returns>
public async Task DeleteKeyPairByName(string deleteKeyName)
{
    try
    {
        await _amazonEc2.DeleteKeyPairAsync(
            new DeleteKeyPairRequest() { KeyName = deleteKeyName });
        File.Delete($"{deleteKeyName}.pem");
    }
    catch (FileNotFoundException)
    {

```

```

        Console.WriteLine($"Key pair {deleteKeyPairName} not found.");
    }
}

/// <summary>
/// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling.
/// The launch template specifies a Bash script in its user data field that
runs after
/// the instance is started. This script installs the Python packages and
starts a Python
/// web server on the instance.
/// </summary>
/// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
/// <param name="instancePolicyPath">The path to a permissions policy to
create and attach to the profile.</param>
/// <returns>The template object.</returns>
public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
{
    await CreateKeyPair(_keyPairName);
    await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName, _instanceProfileName, instancePolicyPath);

    var startServerText = await File.ReadAllTextAsync(startupScriptPath);
    var plainTextBytes = System.Text.Encoding.UTF8.GetBytes(startServerText);

    var amiLatest = await _amazonSsm.GetParameterAsync(
        new GetParameterRequest() { Name = _amiParam });
    var amiId = amiLatest.Parameter.Value;
    var launchTemplateResponse = await _amazonEc2.CreateLaunchTemplateAsync(
        new CreateLaunchTemplateRequest()
        {
            LaunchTemplateName = _launchTemplateName,
            LaunchTemplateData = new RequestLaunchTemplateData()
            {
                InstanceType = _instanceType,
                ImageId = amiId,
                IamInstanceProfile =
                    new
LaunchTemplateIamInstanceProfileSpecificationRequest()
            {

```

```
        Name = _instanceProfileName
    },
    KeyName = _keyPairName,
    UserData = System.Convert.ToBase64String(plainTextBytes)
    }
});
return launchTemplateResponse.LaunchTemplate;
}

/// <summary>
/// Get a list of Availability Zones in the AWS Region of the Amazon EC2
Client.
/// </summary>
/// <returns>A list of availability zones.</returns>
public async Task<List<string>> DescribeAvailabilityZones()
{
    var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
        new DescribeAvailabilityZonesRequest());
    return zoneResponse.AvailabilityZones.Select(z => z.ZoneName).ToList();
}

/// <summary>
/// Create an EC2 Auto Scaling group of a specified size and name.
/// </summary>
/// <param name="groupSize">The size for the group.</param>
/// <param name="groupName">The name for the group.</param>
/// <param name="availabilityZones">The availability zones for the group.</
param>
/// <returns>Async task.</returns>
public async Task CreateGroupOfSize(int groupSize, string groupName,
List<string> availabilityZones)
{
    try
    {
        await _amazonAutoScaling.CreateAutoScalingGroupAsync(
            new CreateAutoScalingGroupRequest()
            {
                AutoScalingGroupName = groupName,
                AvailabilityZones = availabilityZones,
                LaunchTemplate =
                    new
Amazon.AutoScaling.Model.LaunchTemplateSpecification()
                    {
```

```
                LaunchTemplateName = _launchTemplateName,
                Version = "$Default"
            },
            MaxSize = groupSize,
            MinSize = groupSize
        });
        Console.WriteLine($"Created EC2 Auto Scaling group {groupName} with
size {groupSize}.");
    }
    catch (EntityAlreadyExistsException)
    {
        Console.WriteLine($"EC2 Auto Scaling group {groupName} already
exists.");
    }
}

/// <summary>
/// Get the default VPC for the account.
/// </summary>
/// <returns>The default VPC object.</returns>
public async Task<Vpc> GetDefaultVpc()
{
    var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
        new DescribeVpcsRequest()
        {
            Filters = new List<Amazon.EC2.Model.Filter>()
            {
                new ("is-default", new List<string>() { "true" })
            }
        });
    return vpcResponse.Vpcs[0];
}

/// <summary>
/// Get all the subnets for a Vpc in a set of availability zones.
/// </summary>
/// <param name="vpcId">The Id of the Vpc.</param>
/// <param name="availabilityZones">The list of availability zones.</param>
/// <returns>The collection of subnet objects.</returns>
public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
List<string> availabilityZones)
{
    var subnets = new List<Subnet>();
    var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
```

```
        new DescribeSubnetsRequest()
        {
            Filters = new List<Amazon.EC2.Model.Filter>()
            {
                new ("vpc-id", new List<string>() { vpcId}),
                new ("availability-zone", availabilityZones),
                new ("default-for-az", new List<string>() { "true" })
            }
        });

    // Get the entire list using the paginator.
    await foreach (var subnet in subnetPaginator.Subnets)
    {
        subnets.Add(subnet);
    }

    return subnets;
}

/// <summary>
/// Delete a launch template by name.
/// </summary>
/// <param name="templateName">The name of the template to delete.</param>
/// <returns>Async task.</returns>
public async Task DeleteTemplateByName(string templateName)
{
    try
    {
        await _amazonEc2.DeleteLaunchTemplateAsync(
            new DeleteLaunchTemplateRequest()
            {
                LaunchTemplateName = templateName
            });
    }
    catch (AmazonClientException)
    {
        Console.WriteLine($"Unable to delete template {templateName}.");
    }
}

/// <summary>
/// Detaches a role from an instance profile, detaches policies from the
role,
/// and deletes all the resources.
```

```
/// </summary>
/// <param name="profileName">The name of the profile to delete.</param>
/// <param name="roleName">The name of the role to delete.</param>
/// <returns>Async task.</returns>
public async Task DeleteInstanceProfile(string profileName, string roleName)
{
    try
    {
        await _amazonIam.RemoveRoleFromInstanceProfileAsync(
            new RemoveRoleFromInstanceProfileRequest()
            {
                InstanceProfileName = profileName,
                RoleName = roleName
            });
        await _amazonIam.DeleteInstanceProfileAsync(
            new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
        var attachedPolicies = await
_amazonIam.ListAttachedRolePoliciesAsync(
            new ListAttachedRolePoliciesRequest() { RoleName = roleName });
        foreach (var policy in attachedPolicies.AttachedPolicies)
        {
            await _amazonIam.DetachRolePolicyAsync(
                new DetachRolePolicyRequest()
                {
                    RoleName = roleName,
                    PolicyArn = policy.PolicyArn
                });
            // Delete the custom policies only.
            if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
            {
                await _amazonIam.DeletePolicyAsync(
                    new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                    {
                        PolicyArn = policy.PolicyArn
                    });
            }
        }

        await _amazonIam.DeleteRoleAsync(
            new DeleteRoleRequest() { RoleName = roleName });
    }
    catch (NoSuchEntityException)
    {

```

```

        Console.WriteLine($"Instance profile {profileName} does not exist.");
    }
}

/// <summary>
/// Gets data about the instances in an EC2 Auto Scaling group by its group
name.
/// </summary>
/// <param name="group">The name of the auto scaling group.</param>
/// <returns>A collection of instance Ids.</returns>
public async Task<IEnumerable<string>> GetInstancesByGroupName(string group)
{
    var instanceResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
    new DescribeAutoScalingGroupsRequest()
    {
        AutoScalingGroupNames = new List<string>() { group }
    });
    var instanceIds = instanceResponse.AutoScalingGroups.SelectMany(
        g => g.Instances.Select(i => i.InstanceId));
    return instanceIds;
}

/// <summary>
/// Get the instance profile association data for an instance.
/// </summary>
/// <param name="instanceId">The Id of the instance.</param>
/// <returns>Instance profile associations data.</returns>
public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
{
    var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
    new DescribeIamInstanceProfileAssociationsRequest()
    {
        Filters = new List<Amazon.EC2.Model.Filter>()
        {
            new ("instance-id", new List<string>() { instanceId })
        },
    });
    return response.IamInstanceProfileAssociations[0];
}

/// <summary>

```

```
/// Replace the profile associated with a running instance. After the profile
is replaced, the instance
/// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
/// used to restart the Python web server.
/// </summary>
/// <param name="instanceId">The Id of the instance to update.</param>
/// <param name="credsProfileName">The name of the new profile to associate
with the specified instance.</param>
/// <param name="associationId">The Id of the existing profile association
for the instance.</param>
/// <returns>Async task.</returns>
public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
{
    await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
        new ReplaceIamInstanceProfileAssociationRequest()
        {
            AssociationId = associationId,
            IamInstanceProfile = new IamInstanceProfileSpecification()
            {
                Name = credsProfileName
            }
        });
    // Allow time before resetting.
    Thread.Sleep(25000);
    var instanceReady = false;
    var retries = 5;
    while (retries-- > 0 && !instanceReady)
    {
        await _amazonEc2.RebootInstancesAsync(
            new RebootInstancesRequest(new List<string>() { instanceId }));
        Thread.Sleep(10000);

        var instancesPaginator =
            _amazonSsm.Paginators.DescribeInstanceInformation(
                new DescribeInstanceInformationRequest());
        // Get the entire list using the paginator.
        await foreach (var instance in
            instancesPaginator.InstanceInformationList)
        {
            instanceReady = instance.InstanceId == instanceId;
            if (instanceReady)
            {
```

```
        break;
    }
}
}
Console.WriteLine($"Sending restart command to instance {instanceId}");
await _amazonSsm.SendCommandAsync(
    new SendCommandRequest()
    {
        InstanceIds = new List<string>() { instanceId },
        DocumentName = "AWS-RunShellScript",
        Parameters = new Dictionary<string, List<string>>()
        {
            {"commands", new List<string>() { "cd / && sudo python3
server.py 80" }}
        }
    });
Console.WriteLine($"Restarted the web server on instance {instanceId}");
}

/// <summary>
/// Try to terminate an instance by its Id.
/// </summary>
/// <param name="instanceId">The Id of the instance to terminate.</param>
/// <returns>Async task.</returns>
public async Task TryTerminateInstanceById(string instanceId)
{
    var stopping = false;
    Console.WriteLine($"Stopping {instanceId}...");
    while (!stopping)
    {
        try
        {
            await
            _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                new TerminateInstanceInAutoScalingGroupRequest()
                {
                    InstanceId = instanceId,
                    ShouldDecrementDesiredCapacity = false
                });
            stopping = true;
        }
        catch (ScalingActivityInProgressException)
        {

```

```

        Console.WriteLine($"Scaling activity in progress for
{instanceId}. Waiting...");
        Thread.Sleep(10000);
    }
}

/// <summary>
/// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
/// waits and retries until the group is successfully deleted.
/// </summary>
/// <param name="groupName">The name of the group to try to delete.</param>
/// <returns>Async task.</returns>
public async Task TryDeleteGroupByName(string groupName)
{
    var stopped = false;
    while (!stopped)
    {
        try
        {
            await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                new DeleteAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName
                });
            stopped = true;
        }
        catch (Exception e)
            when ((e is ScalingActivityInProgressException)
                || (e is Amazon.AutoScaling.Model.ResourceInUseException))
        {
            Console.WriteLine($"Some instances are still running.
Waiting...");
            Thread.Sleep(10000);
        }
    }
}

/// <summary>
/// Terminate instances and delete the Auto Scaling group by name.
/// </summary>
/// <param name="groupName">The name of the group to delete.</param>
/// <returns>Async task.</returns>

```

```
public async Task TerminateAndDeleteAutoScalingGroupWithName(string
groupName)
{
    var describeGroupsResponse = await
    _amazonAutoScaling.DescribeAutoScalingGroupsAsync(
        new DescribeAutoScalingGroupsRequest()
        {
            AutoScalingGroupNames = new List<string>() { groupName }
        });
    if (describeGroupsResponse.AutoScalingGroups.Any())
    {
        // Update the size to 0.
        await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
            new UpdateAutoScalingGroupRequest()
            {
                AutoScalingGroupName = groupName,
                MinSize = 0
            });
        var group = describeGroupsResponse.AutoScalingGroups[0];
        foreach (var instance in group.Instances)
        {
            await TryTerminateInstanceById(instance.InstanceId);
        }

        await TryDeleteGroupByName(groupName);
    }
    else
    {
        Console.WriteLine($"No groups found with name {groupName}.");
    }
}

/// <summary>
/// Get the default security group for a specified Vpc.
/// </summary>
/// <param name="vpc">The Vpc to search.</param>
/// <returns>The default security group.</returns>
public async Task<SecurityGroup> GetDefaultSecurityGroupForVpc(Vpc vpc)
{
    var groupResponse = await _amazonEc2.DescribeSecurityGroupsAsync(
        new DescribeSecurityGroupsRequest()
        {
            Filters = new List<Amazon.EC2.Model.Filter>()
```

```
        {
            new ("group-name", new List<string>() { "default" }),
            new ("vpc-id", new List<string>() { vpc.VpcId })
        }
    });
    return groupResponse.SecurityGroups[0];
}

/// <summary>
/// Verify the default security group of a Vpc allows ingress from the
calling computer.
/// This can be done by allowing ingress from this computer's IP address.
/// In some situations, such as connecting from a corporate network, you must
instead specify
/// a prefix list Id. You can also temporarily open the port to any IP
address while running this example.
/// If you do, be sure to remove public access when you're done.
/// </summary>
/// <param name="vpc">The group to check.</param>
/// <param name="port">The port to verify.</param>
/// <param name="ipAddress">This computer's IP address.</param>
/// <returns>True if the ip address is allowed on the group.</returns>
public bool VerifyInboundPortForGroup(SecurityGroup group, int port, string
ipAddress)
{
    var portIsOpen = false;
    foreach (var ipPermission in group.IpPermissions)
    {
        if (ipPermission.FromPort == port)
        {
            foreach (var ipRange in ipPermission.Ipv4Ranges)
            {
                var cidr = ipRange.CidrIp;
                if (cidr.StartsWith(ipAddress) || cidr == "0.0.0.0/0")
                {
                    portIsOpen = true;
                }
            }
        }

        if (ipPermission.PrefixListIds.Any())
        {
            portIsOpen = true;
        }
    }
}
```

```
        if (!portIsOpen)
        {
            Console.WriteLine("The inbound rule does not appear to be
open to either this computer's IP\n" +
                                "address, to all IP addresses (0.0.0.0/0),
or to a prefix list ID.");
        }
        else
        {
            break;
        }
    }
}

return portIsOpen;
}

/// <summary>
/// Add an ingress rule to the specified security group that allows access on
the
/// specified port from the specified IP address.
/// </summary>
/// <param name="groupId">The Id of the security group to modify.</param>
/// <param name="port">The port to open.</param>
/// <param name="ipAddress">The IP address to allow access.</param>
/// <returns>Async task.</returns>
public async Task OpenInboundPort(string groupId, int port, string ipAddress)
{
    await _amazonEc2.AuthorizeSecurityGroupIngressAsync(
        new AuthorizeSecurityGroupIngressRequest()
        {
            GroupId = groupId,
            IpPermissions = new List<IpPermission>()
            {
                new IpPermission()
                {
                    FromPort = port,
                    ToPort = port,
                    IpProtocol = "tcp",
                    Ipv4Ranges = new List<IpRange>()
                    {
                        new IpRange() { CidrIp = $"{ipAddress}/32" }
                    }
                }
            }
        }
    );
}
```

```

        }
    });
}

/// <summary>
/// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
/// The
/// </summary>
/// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
/// <param name="targetGroupArn">The Arn for the target group.</param>
/// <returns>Async task.</returns>
public async Task AttachLoadBalancerToGroup(string autoScalingGroupName,
string targetGroupArn)
{
    await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
        new AttachLoadBalancerTargetGroupsRequest()
        {
            AutoScalingGroupName = autoScalingGroupName,
            TargetGroupARNs = new List<string>() { targetGroupArn }
        });
}
}

```

Créez une classe qui englobe les actions Elastic Load Balancing.

```

/// <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);
        }
    }
}
}

```

Créez une classe qui utilise DynamoDB pour simuler un service de recommandation.

```

/// <summary>
/// Encapsulates a DynamoDB table to use as a service that recommends books,
/// movies, and songs.
/// </summary>

```

```
public class Recommendations
{
    private readonly IAmazonDynamoDB _amazonDynamoDb;
    private readonly DynamoDBContext _context;
    private readonly string _tableName;

    public string TableName => _tableName;

    /// <summary>
    /// Constructor for the Recommendations service.
    /// </summary>
    /// <param name="amazonDynamoDb">The injected DynamoDb client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public Recommendations(IAmazonDynamoDB amazonDynamoDb, IConfiguration
configuration)
    {
        _amazonDynamoDb = amazonDynamoDb;
        _context = new DynamoDBContext(_amazonDynamoDb);
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Create the DynamoDb table with a specified name.
    /// </summary>
    /// <param name="tableName">The name for the table.</param>
    /// <returns>True when ready.</returns>
    public async Task<bool> CreateDatabaseWithName(string tableName)
    {
        try
        {
            Console.WriteLine($"Creating table {tableName}...");
            var createRequest = new CreateTableRequest()
            {
                TableName = tableName,
                AttributeDefinitions = new List<AttributeDefinition>()
                {
                    new AttributeDefinition()
                    {
                        AttributeName = "MediaType",
                        AttributeType = ScalarAttributeType.S
                    },
                    new AttributeDefinition()
                    {
                        AttributeName = "ItemId",
```

```
        AttributeType = ScalarAttributeType.N
    }
},
KeySchema = new List<KeySchemaElement>()
{
    new KeySchemaElement()
    {
        AttributeName = "MediaType",
        KeyType = KeyType.HASH
    },
    new KeySchemaElement()
    {
        AttributeName = "ItemId",
        KeyType = KeyType.RANGE
    }
},
ProvisionedThroughput = new ProvisionedThroughput()
{
    ReadCapacityUnits = 5,
    WriteCapacityUnits = 5
}
};
await _amazonDynamoDb.CreateTableAsync(createRequest);

// Wait until the table is ACTIVE and then report success.
Console.WriteLine("\nWaiting for table to become active...");

var request = new DescribeTableRequest
{
    TableName = tableName
};

TableStatus status;
do
{
    Thread.Sleep(2000);

    var describeTableResponse = await
_amazonDynamoDb.DescribeTableAsync(request);
    status = describeTableResponse.Table.TableStatus;

    Console.WriteLine(".");
}
while (status != "ACTIVE");
```

```
        return status == TableStatus.ACTIVE;
    }
    catch (ResourceInUseException)
    {
        Console.WriteLine($"Table {tableName} already exists.");
        return false;
    }
}

/// <summary>
/// Populate the database table with data from a specified path.
/// </summary>
/// <param name="databaseTableName">The name of the table.</param>
/// <param name="recommendationsPath">The path of the recommendations data.</
param>
/// <returns>Async task.</returns>
public async Task PopulateDatabase(string databaseTableName, string
recommendationsPath)
{
    var recommendationsText = await
File.ReadAllTextAsync(recommendationsPath);
    var records =

JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);
    var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

    foreach (var record in records!)
    {
        batchWrite.AddPutItem(record);
    }

    await batchWrite.ExecuteAsync();
}

/// <summary>
/// Delete the recommendation table by name.
/// </summary>
/// <param name="tableName">The name of the recommendation table.</param>
/// <returns>Async task.</returns>
public async Task DestroyDatabaseByName(string tableName)
{
    try
    {
```

```
        await _amazonDynamoDb.DeleteTableAsync(
            new DeleteTableRequest() { TableName = tableName });
        Console.WriteLine($"Table {tableName} was deleted.");
    }
    catch (ResourceNotFoundException)
    {
        Console.WriteLine($"Table {tableName} not found");
    }
}
}
```

Créez une classe qui englobe les actions de Systems Manager.

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

```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for .NET .
 - [AttachLoadBalancerTargetGroups](#)
 - [CreateAutoScalingGroup](#)
 - [CreateInstanceProfile](#)
 - [CreateLaunchTemplate](#)

- [CreateListener](#)
- [CreateLoadBalancer](#)
- [CreateTargetGroup](#)
- [DeleteAutoScalingGroup](#)
- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplacelamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Exécutez un scénario interactif à une invite de commande.

```
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 =
HttpClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
```

```
        HttpResponse response = httpClient.execute(httpGet);
        int statusCode =
response.getStatusLine().getStatusCode();
        System.out.println("HTTP Status Code: " + statusCode);

        // Display the JSON response
        BufferedReader reader = new BufferedReader(
            new
InputStreamReader(response.getEntity().getContent()));
        StringBuilder jsonResponse = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            jsonResponse.append(line);
        }
        reader.close();

        // Print the formatted JSON response.
        System.out.println("Full Response:\n");
        System.out.println(jsonResponse.toString());

        // Close the HTTP client.
        httpClient.close();
    }
    case 1 -> {
        System.out.println("\nChecking the health of load
balancer targets:\n");
        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
        for (TargetHealthDescription target : health) {
            System.out.printf("\tTarget %s on port %d is %s\n",
target.target().id(),
                target.target().port(),
target.targetHealth().stateAsString());
        }
        System.out.println("""
health check to update
                Note that it can take a minute or two for the
                after changes are made.
                """);
    }
    case 2 -> {
        System.out.println("\nOkay, let's move on.");
        System.out.println("-".repeat(88));
    }
}
```

```
        return; // Exit the method when choice is 2
    }
    default -> System.out.println("You must choose a value
between 0-2. Please select again.");
    }

    } catch (java.util.InputMismatchException e) {
        System.out.println("Invalid input. Please select again.");
        scanner.nextLine(); // Clear the input buffer.
    }
}

public static String readFileAsString(String filePath) throws IOException {
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));
    return new String(bytes);
}
}
```

Créez une classe qui englobe les actions Auto Scaling et Amazon EC2.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)

```

```
        .build();
    }
    return ssmClient;
}

private Ec2Client getEc2Client() {
    if (ec2Client == null) {
        ec2Client = Ec2Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ec2Client;
}

private AutoScalingClient getAutoScalingClient() {
    if (autoScalingClient == null) {
        autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return autoScalingClient;
}

/**
 * Terminates and instances in an EC2 Auto Scaling group. After an instance
is
 * terminated, it can no longer be accessed.
 */
public void terminateInstance(String instanceId) {
    TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
TerminateInstanceInAutoScalingGroupRequest
        .builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();

    getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
    System.out.format("Terminated instance %s.", instanceId);
}

/**
 * Replaces the profile associated with a running instance. After the profile
is
```

```
* replaced, the instance is rebooted to ensure that it uses the new profile.
* When
* the instance is ready, Systems Manager is used to restart the Python web
* server.
*/
public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
    throws InterruptedException {
    // Create an IAM instance profile specification.
    software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
    .builder()
    .name(newInstanceProfileName) // Make sure
'newInstanceProfileName' is a valid IAM Instance Profile
    // name.
    .build();

    // Replace the IAM instance profile association for the EC2 instance.
    ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
    .builder()
    .iamInstanceProfile(iamInstanceProfile)
    .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
    .build();

    try {
        getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
        // Handle the response as needed.
    } catch (Ec2Exception e) {
        // Handle exceptions, log, or report the error.
        System.err.println("Error: " + e.getMessage());
    }
    System.out.format("Replaced instance profile for association %s with
profile %s.", profileAssociationId,
        newInstanceProfileName);
    TimeUnit.SECONDS.sleep(15);
    boolean instReady = false;
    int tries = 0;

    // Reboot after 60 seconds
    while (!instReady) {
        if (tries % 6 == 0) {
```

```
        getEc2Client().rebootInstances(RebootInstancesRequest.builder()
            .instanceIds(instanceId)
            .build());
        System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
    }
    tries++;
    try {
        TimeUnit.SECONDS.sleep(10);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
    List<InstanceInformation> instanceInformationList =
informationResponse.getInstanceInformationList();
    for (InstanceInformation info : instanceInformationList) {
        if (info.getInstanceId().equals(instanceId)) {
            instReady = true;
            break;
        }
    }
}

    SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
        .instanceIds(instanceId)
        .documentName("AWS-RunShellScript")
        .parameters(Collections.singletonMap("commands",
            Collections.singletonList("cd / && sudo python3 server.py
80")))
        .build();

    getSSMClient().sendCommand(sendCommandRequest);
    System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
}

    public void openInboundPort(String secGroupId, String port, String ipAddress)
    {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(secGroupId)
            .cidrIp(ipAddress)
```

```
        .fromPort(Integer.parseInt(port))
        .build();

    getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
    System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

/**
 * Detaches a role from an instance profile, detaches policies from the role,
 * and deletes all the resources.
 */
public void deleteInstanceProfile(String roleName, String profileName) {
    try {
        software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
        .builder()
        .instanceProfileName(profileName)
        .build();

        GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
        String name = response.getInstanceProfile().getInstanceProfileName();
        System.out.println(name);

        RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
        .instanceProfileName(profileName)
        .roleName(roleName)
        .build();

        getIAMClient().removeRoleFromInstanceProfile(profileRequest);
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
        .instanceProfileName(profileName)
        .build();

        getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
        System.out.println("Deleted instance profile " + profileName);

        DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
        .roleName(roleName)
        .build();
```

```
        // List attached role policies.
        ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
            .listAttachedRolePolicies(role -> role.roleName(roleName));
        List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
        for (AttachedPolicy attachedPolicy : attachedPolicies) {
            DetachRolePolicyRequest request =
DetachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(attachedPolicy.policyArn())
                .build();

            getIAMClient().detachRolePolicy(request);
            System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
        }

        getIAMClient().deleteRole(deleteRoleRequest);
        System.out.println("Instance profile and role deleted.");

    } catch (IamException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public void deleteTemplate(String templateName) {
    getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
    System.out.format(templateName + " was deleted.");
}

public void deleteAutoScaleGroup(String groupName) {
    DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
        .autoScalingGroupName(groupName)
        .forceDelete(true)
        .build();

    getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
    System.out.println(groupName + " was deleted.");
}
```

```
/*
 * Verify the default security group of the specified VPC allows ingress from
 * this
 * computer. This can be done by allowing ingress from this computer's IP
 * address. In some situations, such as connecting from a corporate network,
you
 * must instead specify a prefix list ID. You can also temporarily open the
port
 * to
 * any IP address while running this example. If you do, be sure to remove
 * public
 * access when you're done.
 *
 */
public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
    boolean portIsOpen = false;
    GroupInfo groupInfo = new GroupInfo();
    try {
        Filter filter = Filter.builder()
            .name("group-name")
            .values("default")
            .build();

        Filter filter1 = Filter.builder()
            .name("vpc-id")
            .values(VPC)
            .build();

        DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
            .filters(filter, filter1)
            .build();

        DescribeSecurityGroupsResponse securityGroupsResponse =
getEc2Client()
            .describeSecurityGroups(securityGroupsRequest);
        String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
        groupInfo.setGroupName(securityGroup);

        for (SecurityGroup secGroup :
securityGroupsResponse.securityGroups()) {
```

```

        System.out.println("Found security group: " +
secGroup.groupId());

        for (IpPermission ipPermission : secGroup.ipPermissions()) {
            if (ipPermission.fromPort() == port) {
                System.out.println("Found inbound rule: " +
ipPermission);
                for (IpRange ipRange : ipPermission.ipRanges()) {
                    String cidrIp = ipRange.cidrIp();
                    if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                        System.out.println(cidrIp + " is applicable");
                        portIsOpen = true;
                    }
                }

                if (!ipPermission.prefixListIds().isEmpty()) {
                    System.out.println("Prefix lList is applicable");
                    portIsOpen = true;
                }

                if (!portIsOpen) {
                    System.out
                        .println("The inbound rule does not appear to
be open to either this computer's IP,"
                                + " all IP addresses (0.0.0.0/0), or
to a prefix list ID.");
                } else {
                    break;
                }
            }
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }

    groupInfo.setPortOpen(portIsOpen);
    return groupInfo;
}

/*
 * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto

```

```
    * Scaling group.
    * The target group specifies how the load balancer forward requests to the
    * instances
    * in the group.
    */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                .autoScalingGroupName(asGroupName)
                .targetGroupARNs(targetGroupARN)
                .build();

            getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

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

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

        // Get availability zones.

        software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

        .map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());
    }
}
```

```
String availabilityZones = String.join(",", availabilityZoneNames);
LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
    .launchTemplateName(templateName)
    .version("$Default")
    .build();

String[] zones = availabilityZones.split(",");
CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
    .launchTemplate(specification)
    .availabilityZones(zones)
    .maxSize(groupSize)
    .minSize(groupSize)
    .autoScalingGroupName(autoScalingGroupName)
    .build();

try {
    getAutoScalingClient().createAutoScalingGroup(groupRequest);
} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
return zones;
}

public String getDefaultVPC() {
    // Define the filter.
    Filter defaultFilter = Filter.builder()
        .name("is-default")
        .values("true")
        .build();

    software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
        .builder()
        .filters(defaultFilter)
        .build();

    DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
    return response.vpcs().get(0).vpcId();
}
```

```
}

// Gets the default subnets in a VPC for a specified list of Availability
Zones.
public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
    List<Subnet> subnets = null;
    Filter vpcFilter = Filter.builder()
        .name("vpc-id")
        .values(vpcId)
        .build();

    Filter azFilter = Filter.builder()
        .name("availability-zone")
        .values(availabilityZones)
        .build();

    Filter defaultForAZ = Filter.builder()
        .name("default-for-az")
        .values("true")
        .build();

    DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
        .filters(vpcFilter, azFilter, defaultForAZ)
        .build();

    DescribeSubnetsResponse response =
getEc2Client().describeSubnets(request);
    subnets = response.subnets();
    return subnets;
}

// Gets data about the instances in the EC2 Auto Scaling group.
public String getBadInstance(String groupName) {
    DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

    DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
    AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
    List<String> instanceIds = autoScalingGroup.instances().stream()
        .map(instance -> instance.instanceId())
        .collect(Collectors.toList());
}
```

```
String[] instanceIdArray = instanceIds.toArray(new String[0]);
for (String instanceId : instanceIdArray) {
    System.out.println("Instance ID: " + instanceId);
    return instanceId;
}
return "";
}

// Gets data about the profile associated with an instance.
public String getInstanceProfile(String instanceId) {
    Filter filter = Filter.builder()
        .name("instance-id")
        .values(instanceId)
        .build();

    DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
        .builder()
        .filters(filter)
        .build();

    DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
        .describeIamInstanceProfileAssociations(associationsRequest);
    return response.iamInstanceProfileAssociations().get(0).associationId();
}

public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
    ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
    ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
    for (Policy policy : listPoliciesResponse.policies()) {
        if (policy.policyName().equals(policyName)) {
            // List the entities (users, groups, roles) that are attached to
the policy.

software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
        .builder()
        .policyArn(policy.arn())
        .build();
```

```
        ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
            .listEntitiesForPolicy(listEntitiesRequest);
        if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
            || !listEntitiesResponse.policyRoles().isEmpty()) {
            // Detach the policy from any entities it is attached to.
            DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
                .policyArn(policy.arn())
                .roleName(roleName) // Specify the name of the IAM
role
                .build();

            getIAMClient().detachRolePolicy(detachPolicyRequest);
            System.out.println("Policy detached from entities.");
        }

        // Now, you can delete the policy.
        DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
            .policyArn(policy.arn())
            .build();

        getIAMClient().deletePolicy(deletePolicyRequest);
        System.out.println("Policy deleted successfully.");
        break;
    }
}

// List the roles associated with the instance profile
ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
    .roleName(roleName)
    .build();

// Detach the roles from the instance profile
ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
    RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
        .instanceProfileName(InstanceProfile)
        .roleName(roleName) // Remove the extra dot here
```

```

        .build();

        getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
        System.out.println("Role " + roleName + " removed from instance
profile " + InstanceProfile);
    }

    // Delete the instance profile after removing all roles
    DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
        .instanceProfileName(InstanceProfile)
        .build();

    getIAMClient().deleteInstanceProfile(r ->
r.instanceProfileName(InstanceProfile));
    System.out.println(InstanceProfile + " Deleted");
    System.out.println("All roles and policies are deleted.");
}
}

```

Créez une classe qui englobe les actions Elastic Load Balancing.

```

public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                .region(Region.US_EAST_1)
                .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String
targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
            .names(targetGroupName)
            .build();
    }
}

```

```
        DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()

        .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
            .build();

        DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }

    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()

            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                .build();

            getLoadBalancerClient().deleteLoadBalancer(
                builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter

                .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
        }
    }
}
```

```
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}

// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
            .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}

// Verify this computer can successfully send a GET request to the load
balancer
// endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws
IOException, InterruptedException {
    boolean success = false;
    int retries = 3;
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to the ELB.
    HttpGet httpGet = new HttpGet("http://" + elbDnsName);
    try {
        while ((!success) && (retries > 0)) {
            // Execute the request and get the response.
            HttpResponse response = httpClient.execute(httpGet);
            int statusCode = response.getStatusLine().getStatusCode();
            System.out.println("HTTP Status Code: " + statusCode);
            if (statusCode == 200) {
                success = true;
            } else {
                retries--;
                System.out.println("Got connection error from load balancer
endpoint, retrying...");
            }
        }
    }
}
```

```
        TimeUnit.SECONDS.sleep(15);
    }
}

} catch (org.apache.http.conn.HttpHostConnectException e) {
    System.out.println(e.getMessage());
}

System.out.println("Status.." + success);
return success;
}

/**
 * Creates an Elastic Load Balancing target group. The target group specifies
 * how
 * the load balancer forward requests to instances in the group and how
instance
 * health is checked.
 */
public String createTargetGroup(String protocol, int port, String vpcId,
String targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
        .name(targetGroupName)
        .protocol(protocol)
        .build();

    CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
    String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
    String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
    System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
    return targetGroupArn;
}

/**
 * Creates an Elastic Load Balancing load balancer that uses the specified
```

```
* subnets
* and forwards requests to the specified target group.
*/
public String createLoadBalancer(List<Subnet> subnetIds, String
targetGroupARN, String lbName, int port,
    String protocol) {
    try {
        List<String> subnetIdStrings = subnetIds.stream()
            .map(Subnet::subnetId)
            .collect(Collectors.toList());

        CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
            .subnets(subnetIdStrings)
            .name(lbName)
            .scheme("internet-facing")
            .build();

        // Create and wait for the load balancer to become available.
        CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
        String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(lbARN)
            .build();

        System.out.println("Waiting for Load Balancer " + lbName + " to
become available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
        Action action = Action.builder()
```

```

        .targetGroupArn(targetGroupARN)
        .type("forward")
        .build();

        CreateListenerRequest listenerRequest =
CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
        .defaultActions(action)
        .port(port)
        .protocol(protocol)
        .defaultActions(action)
        .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
        + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
}

```

Créez une classe qui utilise DynamoDB pour simuler un service de recommandation.

```

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);
}
}
```

Créez une classe qui englobe les actions de Systems Manager.

```
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);  
}  
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [AttachLoadBalancerTargetGroups](#)
 - [CreateAutoScalingGroup](#)
 - [CreateInstanceProfile](#)
 - [CreateLaunchTemplate](#)
 - [CreateListener](#)
 - [CreateLoadBalancer](#)
 - [CreateTargetGroup](#)
 - [DeleteAutoScalingGroup](#)
 - [DeleteInstanceProfile](#)

- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribelamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplacelamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Exécutez un scénario interactif à une invite de commande.

```
#!/usr/bin/env node
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

import {
```

```
Scenario,
  parseScenarioArgs,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * The workflow steps are split into three stages:
 * - deploy
 * - demo
 * - destroy
 *
 * Each of these stages has a corresponding file prefixed with steps-*.
 */
import { deploySteps } from "./steps-deploy.js";
import { demoSteps } from "./steps-demo.js";
import { destroySteps } from "./steps-destroy.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
export const scenarios = {
  // Deploys all resources necessary for the workflow.
  deploy: new Scenario("Resilient Workflow - Deploy", deploySteps, context),
  // Demonstrates how a fragile web service can be made more resilient.
  demo: new Scenario("Resilient Workflow - Demo", demoSteps, context),
  // Destroys the resources created for the workflow.
  destroy: new Scenario("Resilient Workflow - Destroy", destroySteps, context),
};

// Call function if run directly
import { fileURLToPath } from "url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios);
}
```

Créez des étapes pour déployer toutes les ressources.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { join } from "node:path";
import { readFileSync, writeFileSync } from "node:fs";
import axios from "axios";

import {
  BatchWriteItemCommand,
  CreateTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  CreateKeyPairCommand,
  CreateLaunchTemplateCommand,
  DescribeAvailabilityZonesCommand,
  DescribeVpcsCommand,
  DescribeSubnetsCommand,
  DescribeSecurityGroupsCommand,
  AuthorizeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  AttachRolePolicyCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import { SSMClient, GetParameterCommand } from "@aws-sdk/client-ssm";
import {
  CreateAutoScalingGroupCommand,
  AutoScalingClient,
  AttachLoadBalancerTargetGroupsCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  CreateListenerCommand,
  CreateLoadBalancerCommand,
  CreateTargetGroupCommand,
  ElasticLoadBalancingV2Client,
}
```

```
    waitUntilLoadBalancerAvailable,
  } from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH, ROOT } from "./constants.js";
import { initParamsSteps } from "./steps-reset-params.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const deploySteps = [
  new ScenarioOutput("introduction", MESSAGES.introduction, { header: true }),
  new ScenarioInput("confirmDeployment", MESSAGES.confirmDeployment, {
    type: "confirm",
  }),
  new ScenarioAction(
    "handleConfirmDeployment",
    (c) => c.confirmDeployment === false && process.exit(),
  ),
  new ScenarioOutput(
    "creatingTable",
    MESSAGES.creatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("createTable", async () => {
    const client = new DynamoDBClient({});
    await client.send(
      new CreateTableCommand({
        TableName: NAMES.tableName,
        ProvisionedThroughput: {
          ReadCapacityUnits: 5,
          WriteCapacityUnits: 5,
        },
        AttributeDefinitions: [
          {
            AttributeName: "MediaType",
            AttributeType: "S",
          },
          {

```

```

        AttributeName: "ItemId",
        AttributeType: "N",
    },
],
KeySchema: [
    {
        AttributeName: "MediaTypeId",
        KeyType: "HASH",
    },
    {
        AttributeName: "ItemId",
        KeyType: "RANGE",
    },
],
}),
);
await waitUntilTableExists({ client }, { TableName: NAMES.tableName });
}),
new ScenarioOutput(
    "createdTable",
    MESSAGES.createdTable.replace("${TABLE_NAME}", NAMES.tableName),
),
new ScenarioOutput(
    "populatingTable",
    MESSAGES.populatingTable.replace("${TABLE_NAME}", NAMES.tableName),
),
new ScenarioAction("populateTable", () => {
    const client = new DynamoDBClient({});
    /**
     * @type {{ default: import("@aws-sdk/client-dynamodb").PutRequest['Item']
[] }}
    */
    const recommendations = JSON.parse(
        readFileSync(join(RESOURCES_PATH, "recommendations.json")),
    );

    return client.send(
        new BatchWriteItemCommand({
            RequestItems: {
                [NAMES.tableName]: recommendations.map((item) => ({
                    PutRequest: { Item: item },
                })),
            },
        }),
    ),
});

```

```
    );
  }),
  new ScenarioOutput(
    "populatedTable",
    MESSAGES.populatedTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "creatingKeyPair",
    MESSAGES.creatingKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioAction("createKeyPair", async () => {
    const client = new EC2Client({});
    const { KeyMaterial } = await client.send(
      new CreateKeyPairCommand({
        KeyName: NAMES.keyPairName,
      }),
    );
    writeFileSync(`${NAMES.keyPairName}.pem`, KeyMaterial, { mode: 0o600 });
  }),
  new ScenarioOutput(
    "createdKeyPair",
    MESSAGES.createdKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioOutput(
    "creatingInstancePolicy",
    MESSAGES.creatingInstancePolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    ),
  ),
  new ScenarioAction("createInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const {
      Policy: { Arn },
    } = await client.send(
      new CreatePolicyCommand({
        PolicyName: NAMES.instancePolicyName,
        PolicyDocument: readFileSync(
          join(RESOURCES_PATH, "instance_policy.json"),
        ),
      }),
    ),
  }),
  );
state.instancePolicyArn = Arn;
```

```
    }),
    new ScenarioOutput("createdInstancePolicy", (state) =>
      MESSAGES.createdInstancePolicy
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_POLICY_ARN}", state.instancePolicyArn),
    ),
    new ScenarioOutput(
      "creatingInstanceRole",
      MESSAGES.creatingInstanceRole.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      ),
    ),
    new ScenarioAction("createInstanceRole", () => {
      const client = new IAMClient({});
      return client.send(
        new CreateRoleCommand({
          RoleName: NAMES.instanceRoleName,
          AssumeRolePolicyDocument: readFileSync(
            join(ROOT, "assume-role-policy.json"),
          ),
        }),
      ),
    });
  }),
  new ScenarioOutput(
    "createdInstanceRole",
    MESSAGES.createdInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioOutput(
    "attachingPolicyToRole",
    MESSAGES.attachingPolicyToRole
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName)
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName),
  ),
  new ScenarioAction("attachPolicyToRole", async (state) => {
    const client = new IAMClient({});
    await client.send(
      new AttachRolePolicyCommand({
        RoleName: NAMES.instanceRoleName,
        PolicyArn: state.instancePolicyArn,
      }),
    ),
  },
```

```
    );
  }),
  new ScenarioOutput(
    "attachedPolicyToRole",
    MESSAGES.attachedPolicyToRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioOutput(
    "creatingInstanceProfile",
    MESSAGES.creatingInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    ),
  ),
  new ScenarioAction("createInstanceProfile", async (state) => {
    const client = new IAMClient({});
    const {
      InstanceProfile: { Arn },
    } = await client.send(
      new CreateInstanceProfileCommand({
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
    state.instanceProfileArn = Arn;

    await waitUntilInstanceProfileExists(
      { client },
      { InstanceProfileName: NAMES.instanceProfileName },
    );
  }),
  new ScenarioOutput("createdInstanceProfile", (state) =>
    MESSAGES.createdInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_PROFILE_ARN}", state.instanceProfileArn),
  ),
  new ScenarioOutput(
    "addingRoleToInstanceProfile",
    MESSAGES.addingRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioAction("addRoleToInstanceProfile", () => {
    const client = new IAMClient({});
```

```
return client.send(
  new AddRoleToInstanceProfileCommand({
    RoleName: NAMES.instanceRoleName,
    InstanceProfileName: NAMES.instanceProfileName,
  }),
);
}),
new ScenarioOutput(
  "addedRoleToInstanceProfile",
  MESSAGES.addedRoleToInstanceProfile
    .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
    .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
),
...initParamsSteps,
new ScenarioOutput("creatingLaunchTemplate", MESSAGES.creatingLaunchTemplate),
new ScenarioAction("createLaunchTemplate", async () => {
  // snippet-start:[javascript.v3.wkflw.resilient.CreateLaunchTemplate]
  const ssmClient = new SSMClient({});
  const { Parameter } = await ssmClient.send(
    new GetParameterCommand({
      Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
    }),
  );
  const ec2Client = new EC2Client({});
  await ec2Client.send(
    new CreateLaunchTemplateCommand({
      LaunchTemplateName: NAMES.launchTemplateName,
      LaunchTemplateData: {
        InstanceType: "t3.micro",
        ImageId: Parameter.Value,
        IamInstanceProfile: { Name: NAMES.instanceProfileName },
        UserData: readFileSync(
          join(RESOURCES_PATH, "server_startup_script.sh"),
        ).toString("base64"),
        KeyName: NAMES.keyPairName,
      },
    }),
  );
  // snippet-end:[javascript.v3.wkflw.resilient.CreateLaunchTemplate]
});
}),
new ScenarioOutput(
  "createdLaunchTemplate",
  MESSAGES.createdLaunchTemplate.replace(
    "${LAUNCH_TEMPLATE_NAME}",
```

```

    NAMES.launchTemplateName,
  ),
),
new ScenarioOutput(
  "creatingAutoScalingGroup",
  MESSAGES.creatingAutoScalingGroup.replace(
    "${AUTO_SCALING_GROUP_NAME}",
    NAMES.autoScalingGroupName,
  ),
),
new ScenarioAction("createAutoScalingGroup", async (state) => {
  const ec2Client = new EC2Client({});
  const { AvailabilityZones } = await ec2Client.send(
    new DescribeAvailabilityZonesCommand({}),
  );
  state.availabilityZoneNames = AvailabilityZones.map((az) => az.ZoneName);
  const autoScalingClient = new AutoScalingClient({});
  await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
    autoScalingClient.send(
      new CreateAutoScalingGroupCommand({
        AvailabilityZones: state.availabilityZoneNames,
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        LaunchTemplate: {
          LaunchTemplateName: NAMES.launchTemplateName,
          Version: "$Default",
        },
        MinSize: 3,
        MaxSize: 3,
      }),
    ),
  );
}),
new ScenarioOutput(
  "createdAutoScalingGroup",
  /**
   * @param {{ availabilityZoneNames: string[] }} state
   */
  (state) =>
    MESSAGES.createdAutoScalingGroup
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName)
      .replace(
        "${AVAILABILITY_ZONE_NAMES}",
        state.availabilityZoneNames.join(", "),
      ),
),

```

```

    ),
    new ScenarioInput("confirmContinue", MESSAGES.confirmContinue, {
      type: "confirm",
    }),
    new ScenarioOutput("loadBalancer", MESSAGES.loadBalancer),
    new ScenarioOutput("gettingVpc", MESSAGES.gettingVpc),
    new ScenarioAction("getVpc", async (state) => {
      // snippet-start:[javascript.v3.wkflw.resilient.DescribeVpcs]
      const client = new EC2Client({});
      const { Vpcs } = await client.send(
        new DescribeVpcsCommand({
          Filters: [{ Name: "is-default", Values: ["true"] }],
        }),
      );
      // snippet-end:[javascript.v3.wkflw.resilient.DescribeVpcs]
      state.defaultVpc = Vpcs[0].VpcId;
    }),
    new ScenarioOutput("gotVpc", (state) =>
      MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
    ),
    new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
    new ScenarioAction("getSubnets", async (state) => {
      // snippet-start:[javascript.v3.wkflw.resilient.DescribeSubnets]
      const client = new EC2Client({});
      const { Subnets } = await client.send(
        new DescribeSubnetsCommand({
          Filters: [
            { Name: "vpc-id", Values: [state.defaultVpc] },
            { Name: "availability-zone", Values: state.availabilityZoneNames },
            { Name: "default-for-az", Values: ["true"] },
          ],
        }),
      );
      // snippet-end:[javascript.v3.wkflw.resilient.DescribeSubnets]
      state.subnets = Subnets.map((subnet) => subnet.SubnetId);
    }),
    new ScenarioOutput(
      "gotSubnets",
      /**
       * @param {{ subnets: string[] }} state
       */
      (state) =>
        MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
    ),
  ),

```

```
new ScenarioOutput(
  "creatingLoadBalancerTargetGroup",
  MESSAGES.creatingLoadBalancerTargetGroup.replace(
    "${TARGET_GROUP_NAME}",
    NAMES.loadBalancerTargetGroupName,
  ),
),
new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
  // snippet-start:[javascript.v3.wkflw.resilient.CreateTargetGroup]
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new CreateTargetGroupCommand({
      Name: NAMES.loadBalancerTargetGroupName,
      Protocol: "HTTP",
      Port: 80,
      HealthCheckPath: "/healthcheck",
      HealthCheckIntervalSeconds: 10,
      HealthCheckTimeoutSeconds: 5,
      HealthyThresholdCount: 2,
      UnhealthyThresholdCount: 2,
      VpcId: state.defaultVpc,
    }),
  );
  // snippet-end:[javascript.v3.wkflw.resilient.CreateTargetGroup]
  const targetGroup = TargetGroups[0];
  state.targetGroupArn = targetGroup.TargetGroupArn;
  state.targetGroupProtocol = targetGroup.Protocol;
  state.targetGroupPort = targetGroup.Port;
}),
new ScenarioOutput(
  "createdLoadBalancerTargetGroup",
  MESSAGES.createdLoadBalancerTargetGroup.replace(
    "${TARGET_GROUP_NAME}",
    NAMES.loadBalancerTargetGroupName,
  ),
),
new ScenarioOutput(
  "creatingLoadBalancer",
  MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
),
new ScenarioAction("createLoadBalancer", async (state) => {
  // snippet-start:[javascript.v3.wkflw.resilient.CreateLoadBalancer]
  const client = new ElasticLoadBalancingV2Client({});
  const { LoadBalancers } = await client.send(
```

```
    new CreateLoadBalancerCommand({
      Name: NAMES.loadBalancerName,
      Subnets: state.subnets,
    }),
  );
state.loadBalancerDns = LoadBalancers[0].DNSName;
state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
await waitUntilLoadBalancerAvailable(
  { client },
  { Names: [NAMES.loadBalancerName] },
);
// snippet-end:[javascript.v3.wkflw.resilient.CreateLoadBalancer]
}),
new ScenarioOutput("createdLoadBalancer", (state) =>
  MESSAGES.createdLoadBalancer
    .replace("${LB_NAME}", NAMES.loadBalancerName)
    .replace("${DNS_NAME}", state.loadBalancerDns),
),
new ScenarioOutput(
  "creatingListener",
  MESSAGES.creatingLoadBalancerListener
    .replace("${LB_NAME}", NAMES.loadBalancerName)
    .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName),
),
new ScenarioAction("createListener", async (state) => {
  // snippet-start:[javascript.v3.wkflw.resilient.CreateListener]
  const client = new ElasticLoadBalancingV2Client({});
  const { Listeners } = await client.send(
    new CreateListenerCommand({
      LoadBalancerArn: state.loadBalancerArn,
      Protocol: state.targetGroupProtocol,
      Port: state.targetGroupPort,
      DefaultActions: [
        { Type: "forward", TargetGroupArn: state.targetGroupArn },
      ],
    }),
  );
  // snippet-end:[javascript.v3.wkflw.resilient.CreateListener]
  const listener = Listeners[0];
  state.loadBalancerListenerArn = listener.ListenerArn;
}),
new ScenarioOutput("createdListener", (state) =>
  MESSAGES.createdLoadBalancerListener.replace(
    "${LB_LISTENER_ARN}",
```

```

    state.loadBalancerListenerArn,
  ),
),
new ScenarioOutput(
  "attachingLoadBalancerTargetGroup",
  MESSAGES.attachingLoadBalancerTargetGroup
    .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName)
    .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName),
),
new ScenarioAction("attachLoadBalancerTargetGroup", async (state) => {
  // snippet-start:[javascript.v3.wkflw.resilient.AttachTargetGroup]
  const client = new AutoScalingClient({});
  await client.send(
    new AttachLoadBalancerTargetGroupsCommand({
      AutoScalingGroupName: NAMES.autoScalingGroupName,
      TargetGroupARNs: [state.targetGroupArn],
    }),
  );
  // snippet-end:[javascript.v3.wkflw.resilient.AttachTargetGroup]
}),
new ScenarioOutput(
  "attachedLoadBalancerTargetGroup",
  MESSAGES.attachedLoadBalancerTargetGroup,
),
new ScenarioOutput("verifyingInboundPort", MESSAGES.verifyingInboundPort),
new ScenarioAction(
  "verifyInboundPort",
  /**
   *
   * @param {{ defaultSecurityGroup: import('@aws-sdk/client-ec2').SecurityGroup}} state
   */
  async (state) => {
    const client = new EC2Client({});
    const { SecurityGroups } = await client.send(
      new DescribeSecurityGroupsCommand({
        Filters: [{ Name: "group-name", Values: ["default"] }],
      }),
    );
    if (!SecurityGroups) {
      state.verifyInboundPortError = new Error(MESSAGES.noSecurityGroups);
    }
    state.defaultSecurityGroup = SecurityGroups[0];
  }
);

```

```
/**
 * @type {string}
 */
const ipResponse = (await axios.get("http://checkip.amazonaws.com")).data;
state.myIp = ipResponse.trim();
const myIpRules = state.defaultSecurityGroup.IpPermissions.filter(
  ({ IpRanges }) =>
    IpRanges.some(
      ({ CidrIp }) =>
        CidrIp.startsWith(state.myIp) || CidrIp === "0.0.0.0/0",
    ),
)
  .filter(({ IpProtocol }) => IpProtocol === "tcp")
  .filter(({ FromPort }) => FromPort === 80);

state.myIpRules = myIpRules;
},
),
new ScenarioOutput(
  "verifiedInboundPort",
  /**
   * @param {{ myIpRules: any[] }} state
   */
  (state) => {
    if (state.myIpRules.length > 0) {
      return MESSAGES.foundIpRules.replace(
        "${IP_RULES}",
        JSON.stringify(state.myIpRules, null, 2),
      );
    } else {
      return MESSAGES.noIpRules;
    }
  },
),
new ScenarioInput(
  "shouldAddInboundRule",
  /**
   * @param {{ myIpRules: any[] }} state
   */
  (state) => {
    if (state.myIpRules.length > 0) {
      return false;
    } else {
      return MESSAGES.noIpRules;
    }
  },
),

```

```

    }
  },
  { type: "confirm" },
),
new ScenarioAction(
  "addInboundRule",
  /**
   * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup }} state
   */
  async (state) => {
    if (!state.shouldAddInboundRule) {
      return;
    }

    const client = new EC2Client({});
    await client.send(
      new AuthorizeSecurityGroupIngressCommand({
        GroupId: state.defaultSecurityGroup.GroupId,
        CidrIp: `${state.myIp}/32`,
        FromPort: 80,
        ToPort: 80,
        IpProtocol: "tcp",
      }),
    );
  },
),
new ScenarioOutput("addedInboundRule", (state) => {
  if (state.shouldAddInboundRule) {
    return MESSAGES.addedInboundRule.replace("${IP_ADDRESS}", state.myIp);
  } else {
    return false;
  }
}),
new ScenarioOutput("verifyingEndpoint", (state) =>
  MESSAGES.verifyingEndpoint.replace("${DNS_NAME}", state.loadBalancerDns),
),
new ScenarioAction("verifyEndpoint", async (state) => {
  try {
    const response = await retry({ intervalInMs: 2000, maxRetries: 30 }, () =>
      axios.get(`http://${state.loadBalancerDns}`),
    );
    state.endpointResponse = JSON.stringify(response.data, null, 2);
  } catch (e) {

```

```
        state.verifyEndpointError = e;
    }
  })),
  new ScenarioOutput("verifiedEndpoint", (state) => {
    if (state.verifyEndpointError) {
      console.error(state.verifyEndpointError);
    } else {
      return MESSAGES.verifiedEndpoint.replace(
        "${ENDPOINT_RESPONSE}",
        state.endpointResponse,
      );
    }
  })),
];
```

Créez des étapes pour exécuter la démo.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { readFileSync } from "node:fs";
import { join } from "node:path";

import axios from "axios";

import {
  DescribeTargetGroupsCommand,
  DescribeTargetHealthCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
import {
  DescribeInstanceInformationCommand,
  PutParameterCommand,
  SSMClient,
  SendCommandCommand,
} from "@aws-sdk/client-ssm";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
```

```
    waitUntilInstanceProfileExists,
  } from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DescribeAutoScalingGroupsCommand,
  TerminateInstanceInAutoScalingGroupCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  DescribeIamInstanceProfileAssociationsCommand,
  EC2Client,
  RebootInstancesCommand,
  ReplaceIamInstanceProfileAssociationCommand,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

const getRecommendation = new ScenarioAction(
  "getRecommendation",
  async (state) => {
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    if (loadBalancer) {
      state.loadBalancerDnsName = loadBalancer.DNSName;
      try {
        state.recommendation = (
          await axios.get(`http://${state.loadBalancerDnsName}`)
        ).data;
      } catch (e) {
        state.recommendation = e instanceof Error ? e.message : e;
      }
    } else {
      throw new Error(MESSAGES.demoFindLoadBalancerError);
    }
  },
);

const getRecommendationResult = new ScenarioOutput(
```

```

    "getRecommendationResult",
    (state) =>
      `Recommendation:\n${JSON.stringify(state.recommendation, null, 2)}`,
    { preformatted: true },
  );

const getHealthCheck = new ScenarioAction("getHealthCheck", async (state) => {
  // snippet-start:[javascript.v3.wkflw.resilient.DescribeTargetGroups]
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );
  // snippet-end:[javascript.v3.wkflw.resilient.DescribeTargetGroups]

  // snippet-start:[javascript.v3.wkflw.resilient.DescribeTargetHealth]
  const { TargetHealthDescriptions } = await client.send(
    new DescribeTargetHealthCommand({
      TargetGroupArn: TargetGroups[0].TargetGroupArn,
    }),
  );
  // snippet-end:[javascript.v3.wkflw.resilient.DescribeTargetHealth]
  state.targetHealthDescriptions = TargetHealthDescriptions;
});

const getHealthCheckResult = new ScenarioOutput(
  "getHealthCheckResult",
  /**
   * @param {{ targetHealthDescriptions: import('@aws-sdk/client-elastic-load-balancing-v2').TargetHealthDescription[][]} state
   */
  (state) => {
    const status = state.targetHealthDescriptions
      .map((th) => `${th.Target.Id}: ${th.TargetHealth.State}`)
      .join("\n");
    return `Health check:\n${status}`;
  },
  { preformatted: true },
);

const loadBalancerLoop = new ScenarioAction(
  "loadBalancerLoop",
  getRecommendation.action,

```

```
{
  whileConfig: {
    whileFn: ({ loadBalancerCheck }) => loadBalancerCheck,
    input: new ScenarioInput(
      "loadBalancerCheck",
      MESSAGES.demoLoadBalancerCheck,
      {
        type: "confirm",
      },
    ),
    output: getRecommendationResult,
  },
},
);

const healthCheckLoop = new ScenarioAction(
  "healthCheckLoop",
  getHealthCheck.action,
  {
    whileConfig: {
      whileFn: ({ healthCheck }) => healthCheck,
      input: new ScenarioInput("healthCheck", MESSAGES.demoHealthCheck, {
        type: "confirm",
      }),
      output: getHealthCheckResult,
    },
  },
);

const statusSteps = [
  getRecommendation,
  getRecommendationResult,
  getHealthCheck,
  getHealthCheckResult,
];

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[][]}
 */
export const demoSteps = [
  new ScenarioOutput("header", MESSAGES.demoHeader, { header: true }),
  new ScenarioOutput("sanityCheck", MESSAGES.demoSanityCheck),
  ...statusSteps,
  new ScenarioInput(
```

```
"brokenDependencyConfirmation",
MESSAGES.demoBrokenDependencyConfirmation,
{ type: "confirm" },
),
new ScenarioAction("brokenDependency", async (state) => {
  if (!state.brokenDependencyConfirmation) {
    process.exit();
  } else {
    const client = new SSMClient({});
    state.badTableName = `fake-table-${Date.now()}`;
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: state.badTableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }
}),
new ScenarioOutput("testBrokenDependency", (state) =>
  MESSAGES.demoTestBrokenDependency.replace(
    "${TABLE_NAME}",
    state.badTableName,
  ),
),
...statusSteps,
new ScenarioInput(
  "staticResponseConfirmation",
  MESSAGES.demoStaticResponseConfirmation,
  { type: "confirm" },
),
new ScenarioAction("staticResponse", async (state) => {
  if (!state.staticResponseConfirmation) {
    process.exit();
  } else {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmFailureResponseKey,
        Value: "static",
        Overwrite: true,
        Type: "String",
      }),
    ),
  }
}),
```

```

    );
  }
}),
new ScenarioOutput("testStaticResponse", MESSAGES.demoTestStaticResponse),
...statusSteps,
new ScenarioInput(
  "badCredentialsConfirmation",
  MESSAGES.demoBadCredentialsConfirmation,
  { type: "confirm" },
),
new ScenarioAction("badCredentialsExit", (state) => {
  if (!state.badCredentialsConfirmation) {
    process.exit();
  }
}),
new ScenarioAction("fixDynamoDBName", async () => {
  const client = new SSMClient({});
  await client.send(
    new PutParameterCommand({
      Name: NAMES.ssmTableNameKey,
      Value: NAMES.tableName,
      Overwrite: true,
      Type: "String",
    }),
  ),
);
}),
new ScenarioAction(
  "badCredentials",
  /**
   * @param {{ targetInstance: import('@aws-sdk/client-auto-
scaling').Instance }} state
   */
  async (state) => {
    await createSsmOnlyInstanceProfile();
    const autoScalingClient = new AutoScalingClient({});
    const { AutoScalingGroups } = await autoScalingClient.send(
      new DescribeAutoScalingGroupsCommand({
        AutoScalingGroupNames: [NAMES.autoScalingGroupName],
      }),
    );
  });
  state.targetInstance = AutoScalingGroups[0].Instances[0];
  // snippet-start:
[javascript.v3.wkflw.resilient.DescribeIamInstanceProfileAssociations]
  const ec2Client = new EC2Client({});

```

```

const { IamInstanceProfileAssociations } = await ec2Client.send(
  new DescribeIamInstanceProfileAssociationsCommand({
    Filters: [
      { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
    ],
  }),
);
// snippet-end:
[javascript.v3.wkflw.resilient.DescribeIamInstanceProfileAssociations]
state.instanceProfileAssociationId =
  IamInstanceProfileAssociations[0].AssociationId;
// snippet-start:
[javascript.v3.wkflw.resilient.ReplaceIamInstanceProfileAssociation]
await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
  ec2Client.send(
    new ReplaceIamInstanceProfileAssociationCommand({
      AssociationId: state.instanceProfileAssociationId,
      IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
    }),
  ),
);
// snippet-end:
[javascript.v3.wkflw.resilient.ReplaceIamInstanceProfileAssociation]

await ec2Client.send(
  new RebootInstancesCommand({
    InstanceIds: [state.targetInstance.InstanceId],
  }),
);

const ssmClient = new SSMClient({});
await retry({ intervalInMs: 20000, maxRetries: 15 }, async () => {
  const { InstanceInformationList } = await ssmClient.send(
    new DescribeInstanceInformationCommand({}),
  );

  const instance = InstanceInformationList.find(
    (info) => info.InstanceId === state.targetInstance.InstanceId,
  );

  if (!instance) {
    throw new Error("Instance not found.");
  }
});

```

```

    await ssmClient.send(
      new SendCommandCommand({
        InstanceIds: [state.targetInstance.InstanceId],
        DocumentName: "AWS-RunShellScript",
        Parameters: { commands: ["cd / && sudo python3 server.py 80"] },
      }),
    );
  },
),
new ScenarioOutput(
  "testBadCredentials",
  /**
   * @param {{ targetInstance: import('@aws-sdk/client-
   ssm').InstanceInformation}} state
   */
  (state) =>
    MESSAGES.demoTestBadCredentials.replace(
      "${INSTANCE_ID}",
      state.targetInstance.InstanceId,
    ),
),
loadBalancerLoop,
new ScenarioInput(
  "deepHealthCheckConfirmation",
  MESSAGES.demoDeepHealthCheckConfirmation,
  { type: "confirm" },
),
new ScenarioAction("deepHealthCheckExit", (state) => {
  if (!state.deepHealthCheckConfirmation) {
    process.exit();
  }
}),
new ScenarioAction("deepHealthCheck", async () => {
  const client = new SSMClient({});
  await client.send(
    new PutParameterCommand({
      Name: NAMES.ssmHealthCheckKey,
      Value: "deep",
      Overwrite: true,
      Type: "String",
    }),
  );
}),
}),

```

```
new ScenarioOutput("testDeepHealthCheck", MESSAGES.demoTestDeepHealthCheck),
healthCheckLoop,
loadBalancerLoop,
new ScenarioInput(
  "killInstanceConfirmation",
  /**
   * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
   */
  (state) =>
    MESSAGES.demoKillInstanceConfirmation.replace(
      "${INSTANCE_ID}",
      state.targetInstance.InstanceId,
    ),
  { type: "confirm" },
),
new ScenarioAction("killInstanceExit", (state) => {
  if (!state.killInstanceConfirmation) {
    process.exit();
  }
}),
new ScenarioAction(
  "killInstance",
  /**
   * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
   */
  async (state) => {
    const client = new AutoScalingClient({});
    await client.send(
      new TerminateInstanceInAutoScalingGroupCommand({
        InstanceId: state.targetInstance.InstanceId,
        ShouldDecrementDesiredCapacity: false,
      }),
    );
  },
),
new ScenarioOutput("testKillInstance", MESSAGES.demoTestKillInstance),
healthCheckLoop,
loadBalancerLoop,
new ScenarioInput("failOpenConfirmation", MESSAGES.demoFailOpenConfirmation, {
  type: "confirm",
}),
new ScenarioAction("failOpenExit", (state) => {
```

```
    if (!state.failOpenConfirmation) {
      process.exit();
    }
  })),
  new ScenarioAction("failOpen", () => {
    const client = new SSMClient({});
    return client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: `fake-table-${Date.now()}`,
        Overwrite: true,
        Type: "String",
      }),
    );
  })),
  new ScenarioOutput("testFailOpen", MESSAGES.demoFailOpenTest),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "resetTableConfirmation",
    MESSAGES.demoResetTableConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("resetTableExit", (state) => {
    if (!state.resetTableConfirmation) {
      process.exit();
    }
  })),
  new ScenarioAction("resetTable", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  })),
  new ScenarioOutput("testResetTable", MESSAGES.demoTestResetTable),
  healthCheckLoop,
  loadBalancerLoop,
];
```

```
async function createSsmOnlyInstanceProfile() {
  const iamClient = new IAMClient({});
  const { Policy } = await iamClient.send(
    new CreatePolicyCommand({
      PolicyName: NAMES.ssmOnlyPolicyName,
      PolicyDocument: readFileSync(
        join(RESOURCES_PATH, "ssm_only_policy.json"),
      ),
    }),
  );
  await iamClient.send(
    new CreateRoleCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      AssumeRolePolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Principal: { Service: "ec2.amazonaws.com" },
            Action: "sts:AssumeRole",
          },
        ],
      }),
    ));
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: Policy.Arn,
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
    }),
  );
  // snippet-start:[javascript.v3.wkflw.resilient.CreateInstanceProfile]
  const { InstanceProfile } = await iamClient.send(
    new CreateInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    }),
  );
  await waitUntilInstanceProfileExists(
```

```

    { client: iamClient },
    { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
  );
// snippet-end:[javascript.v3.wkflw.resilient.CreateInstanceProfile]
await iamClient.send(
  new AddRoleToInstanceProfileCommand({
    InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    RoleName: NAMES.ssmOnlyRoleName,
  })),
);

return InstanceProfile;
}

```

Créez des étapes pour déployer toutes les ressources.

```

// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { unlinkSync } from "node:fs";

import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  DeleteKeyPairCommand,
  DeleteLaunchTemplateCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  DeleteInstanceProfileCommand,
  RemoveRoleFromInstanceProfileCommand,
  DeletePolicyCommand,
  DeleteRoleCommand,
  DetachRolePolicyCommand,
  paginateListPolicies,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DeleteAutoScalingGroupCommand,
  TerminateInstanceInAutoScalingGroupCommand,
  UpdateAutoScalingGroupCommand,
  paginateDescribeAutoScalingGroups,
} from "@aws-sdk/client-auto-scaling";

```

```
import {
  DeleteLoadBalancerCommand,
  DeleteTargetGroupCommand,
  DescribeTargetGroupsCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const destroySteps = [
  new ScenarioInput("destroy", MESSAGES.destroy, { type: "confirm" }),
  new ScenarioAction(
    "abort",
    (state) => state.destroy === false && process.exit(),
  ),
  new ScenarioAction("deleteTable", async (c) => {
    try {
      const client = new DynamoDBClient({});
      await client.send(new DeleteTableCommand({ TableName: NAMES.tableName }));
    } catch (e) {
      c.deleteTableError = e;
    }
  }),
  new ScenarioOutput("deleteTableResult", (state) => {
    if (state.deleteTableError) {
      console.error(state.deleteTableError);
      return MESSAGES.deleteTableError.replace(
        "${TABLE_NAME}",
        NAMES.tableName,
      );
    } else {
      return MESSAGES.deletedTable.replace("${TABLE_NAME}", NAMES.tableName);
    }
  })
];
```

```
    }),
    new ScenarioAction("deleteKeyPair", async (state) => {
      try {
        const client = new EC2Client({});
        await client.send(
          new DeleteKeyPairCommand({ KeyName: NAMES.keyPairName }),
        );
        unlinkSync(`${NAMES.keyPairName}.pem`);
      } catch (e) {
        state.deleteKeyPairError = e;
      }
    }),
    new ScenarioOutput("deleteKeyPairResult", (state) => {
      if (state.deleteKeyPairError) {
        console.error(state.deleteKeyPairError);
        return MESSAGES.deleteKeyPairError.replace(
          "${KEY_PAIR_NAME}",
          NAMES.keyPairName,
        );
      } else {
        return MESSAGES.deletedKeyPair.replace(
          "${KEY_PAIR_NAME}",
          NAMES.keyPairName,
        );
      }
    }),
    new ScenarioAction("detachPolicyFromRole", async (state) => {
      try {
        const client = new IAMClient({});
        const policy = await findPolicy(NAMES.instancePolicyName);

        if (!policy) {
          state.detachPolicyFromRoleError = new Error(
            `Policy ${NAMES.instancePolicyName} not found.`,
          );
        } else {
          await client.send(
            new DetachRolePolicyCommand({
              RoleName: NAMES.instanceRoleName,
              PolicyArn: policy.Arn,
            }),
          );
        }
      } catch (e) {
```

```
    state.detachPolicyFromRoleError = e;
  }
}),
new ScenarioOutput("detachedPolicyFromRole", (state) => {
  if (state.detachPolicyFromRoleError) {
    console.error(state.detachPolicyFromRoleError);
    return MESSAGES.detachPolicyFromRoleError
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  } else {
    return MESSAGES.detachedPolicyFromRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }
}),
new ScenarioAction("deleteInstancePolicy", async (state) => {
  const client = new IAMClient({});
  const policy = await findPolicy(NAMES.instancePolicyName);

  if (!policy) {
    state.deletePolicyError = new Error(
      `Policy ${NAMES.instancePolicyName} not found.`
    );
  } else {
    return client.send(
      new DeletePolicyCommand({
        PolicyArn: policy.Arn,
      })
    );
  }
}),
new ScenarioOutput("deletePolicyResult", (state) => {
  if (state.deletePolicyError) {
    console.error(state.deletePolicyError);
    return MESSAGES.deletePolicyError.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    );
  } else {
    return MESSAGES.deletedPolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    );
  }
})
```

```
   )),
    new ScenarioAction("removeRoleFromInstanceProfile", async (state) => {
      try {
        const client = new IAMClient({});
        await client.send(
          new RemoveRoleFromInstanceProfileCommand({
            RoleName: NAMES.instanceRoleName,
            InstanceProfileName: NAMES.instanceProfileName,
          }),
        );
      } catch (e) {
        state.removeRoleFromInstanceProfileError = e;
      }
    }),
    new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
      if (state.removeRoleFromInstanceProfile) {
        console.error(state.removeRoleFromInstanceProfileError);
        return MESSAGES.removeRoleFromInstanceProfileError
          .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
          .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
      } else {
        return MESSAGES.removedRoleFromInstanceProfile
          .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
          .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
      }
    }),
    new ScenarioAction("deleteInstanceRole", async (state) => {
      try {
        const client = new IAMClient({});
        await client.send(
          new DeleteRoleCommand({
            RoleName: NAMES.instanceRoleName,
          }),
        );
      } catch (e) {
        state.deleteInstanceRoleError = e;
      }
    }),
    new ScenarioOutput("deleteInstanceRoleResult", (state) => {
      if (state.deleteInstanceRoleError) {
        console.error(state.deleteInstanceRoleError);
        return MESSAGES.deleteInstanceRoleError.replace(
          "${INSTANCE_ROLE_NAME}",
          NAMES.instanceRoleName,
        );
      }
    })
  ],
);
```

```
    );
  } else {
    return MESSAGES.deletedInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    );
  }
}),
new ScenarioAction("deleteInstanceProfile", async (state) => {
  try {
    // snippet-start:[javascript.v3.wkflw.resilient.DeleteInstanceProfile]
    const client = new IAMClient({});
    await client.send(
      new DeleteInstanceProfileCommand({
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
    // snippet-end:[javascript.v3.wkflw.resilient.DeleteInstanceProfile]
  } catch (e) {
    state.deleteInstanceProfileError = e;
  }
}),
new ScenarioOutput("deleteInstanceProfileResult", (state) => {
  if (state.deleteInstanceProfileError) {
    console.error(state.deleteInstanceProfileError);
    return MESSAGES.deleteInstanceProfileError.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    );
  } else {
    return MESSAGES.deletedInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    );
  }
}),
new ScenarioAction("deleteAutoScalingGroup", async (state) => {
  try {
    await terminateGroupInstances(NAMES.autoScalingGroupName);
    await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
      await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
    });
  } catch (e) {
    state.deleteAutoScalingGroupError = e;
  }
});
```

```
    }
  })),
  new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {
    if (state.deleteAutoScalingGroupError) {
      console.error(state.deleteAutoScalingGroupError);
      return MESSAGES.deleteAutoScalingGroupError.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    } else {
      return MESSAGES.deletedAutoScalingGroup.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    }
  })),
  new ScenarioAction("deleteLaunchTemplate", async (state) => {
    const client = new EC2Client({});
    try {
      // snippet-start:[javascript.v3.wkflw.resilient.DeleteLaunchTemplate]
      await client.send(
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
      );
      // snippet-end:[javascript.v3.wkflw.resilient.DeleteLaunchTemplate]
    } catch (e) {
      state.deleteLaunchTemplateError = e;
    }
  })),
  new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
    if (state.deleteLaunchTemplateError) {
      console.error(state.deleteLaunchTemplateError);
      return MESSAGES.deleteLaunchTemplateError.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    } else {
      return MESSAGES.deletedLaunchTemplate.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    }
  })),
}
```

```
new ScenarioAction("deleteLoadBalancer", async (state) => {
  try {
    // snippet-start:[javascript.v3.wkflw.resilient.DeleteLoadBalancer]
    const client = new ElasticLoadBalancingV2Client({});
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    await client.send(
      new DeleteLoadBalancerCommand({
        LoadBalancerArn: loadBalancer.LoadBalancerArn,
      }),
    );
    await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
      const lb = await findLoadBalancer(NAMES.loadBalancerName);
      if (lb) {
        throw new Error("Load balancer still exists.");
      }
    });
    // snippet-end:[javascript.v3.wkflw.resilient.DeleteLoadBalancer]
  } catch (e) {
    state.deleteLoadBalancerError = e;
  }
}),
new ScenarioOutput("deleteLoadBalancerResult", (state) => {
  if (state.deleteLoadBalancerError) {
    console.error(state.deleteLoadBalancerError);
    return MESSAGES.deleteLoadBalancerError.replace(
      "${LB_NAME}",
      NAMES.loadBalancerName,
    );
  } else {
    return MESSAGES.deletedLoadBalancer.replace(
      "${LB_NAME}",
      NAMES.loadBalancerName,
    );
  }
}),
new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
  // snippet-start:[javascript.v3.wkflw.resilient.DeleteTargetGroup]
  const client = new ElasticLoadBalancingV2Client({});
  try {
    const { TargetGroups } = await client.send(
      new DescribeTargetGroupsCommand({
        Names: [NAMES.loadBalancerTargetGroupName],
      }),
    );
  }
});
```

```
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      client.send(
        new DeleteTargetGroupCommand({
          TargetGroupArn: TargetGroups[0].TargetGroupArn,
        }),
      ),
    );
  } catch (e) {
    state.deleteLoadBalancerTargetGroupError = e;
  }
  // snippet-end:[javascript.v3.wkflw.resilient.DeleteTargetGroup]
}),
new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
  if (state.deleteLoadBalancerTargetGroupError) {
    console.error(state.deleteLoadBalancerTargetGroupError);
    return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    );
  } else {
    return MESSAGES.deletedLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    );
  }
}),
new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
  try {
    const client = new IAMClient({});
    await client.send(
      new RemoveRoleFromInstanceProfileCommand({
        InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
        RoleName: NAMES.ssmOnlyRoleName,
      }),
    );
  } catch (e) {
    state.detachSsmOnlyRoleFromProfileError = e;
  }
}),
new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
  if (state.detachSsmOnlyRoleFromProfileError) {
    console.error(state.detachSsmOnlyRoleFromProfileError);
    return MESSAGES.detachSsmOnlyRoleFromProfileError
```

```

        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    } else {
        return MESSAGES.detachedSsmOnlyRoleFromProfile
            .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
            .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    }
}),
new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
    try {
        const iamClient = new IAMClient({});
        const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
        await iamClient.send(
            new DetachRolePolicyCommand({
                RoleName: NAMES.ssmOnlyRoleName,
                PolicyArn: ssmOnlyPolicy.Arn,
            })),
        );
    } catch (e) {
        state.detachSsmOnlyCustomRolePolicyError = e;
    }
}),
new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
    if (state.detachSsmOnlyCustomRolePolicyError) {
        console.error(state.detachSsmOnlyCustomRolePolicyError);
        return MESSAGES.detachSsmOnlyCustomRolePolicyError
            .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
            .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    } else {
        return MESSAGES.detachedSsmOnlyCustomRolePolicy
            .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
            .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    }
}),
new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
    try {
        const iamClient = new IAMClient({});
        await iamClient.send(
            new DetachRolePolicyCommand({
                RoleName: NAMES.ssmOnlyRoleName,
                PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
            })),
        );
    } catch (e) {

```

```
    state.detachSsmOnlyAWSRolePolicyError = e;
  }
}),
new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
  if (state.detachSsmOnlyAWSRolePolicyError) {
    console.error(state.detachSsmOnlyAWSRolePolicyError);
    return MESSAGES.detachSsmOnlyAWSRolePolicyError
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
  } else {
    return MESSAGES.detachedSsmOnlyAWSRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
  }
}),
new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {
  try {
    const iamClient = new IAMClient({});
    await iamClient.send(
      new DeleteInstanceProfileCommand({
        InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
      }),
    );
  } catch (e) {
    state.deleteSsmOnlyInstanceProfileError = e;
  }
}),
new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
  if (state.deleteSsmOnlyInstanceProfileError) {
    console.error(state.deleteSsmOnlyInstanceProfileError);
    return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.ssmOnlyInstanceProfileName,
    );
  } else {
    return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.ssmOnlyInstanceProfileName,
    );
  }
}),
new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
  try {
    const iamClient = new IAMClient({});
```

```
const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
await iamClient.send(
  new DeletePolicyCommand({
    PolicyArn: ssmOnlyPolicy.Arn,
  }),
);
} catch (e) {
  state.deleteSsmOnlyPolicyError = e;
}
}),
new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
  if (state.deleteSsmOnlyPolicyError) {
    console.error(state.deleteSsmOnlyPolicyError);
    return MESSAGES.deleteSsmOnlyPolicyError.replace(
      "${POLICY_NAME}",
      NAMES.ssmOnlyPolicyName,
    );
  } else {
    return MESSAGES.deletedSsmOnlyPolicy.replace(
      "${POLICY_NAME}",
      NAMES.ssmOnlyPolicyName,
    );
  }
}),
new ScenarioAction("deleteSsmOnlyRole", async (state) => {
  try {
    const iamClient = new IAMClient({});
    await iamClient.send(
      new DeleteRoleCommand({
        RoleName: NAMES.ssmOnlyRoleName,
      }),
    );
  } catch (e) {
    state.deleteSsmOnlyRoleError = e;
  }
}),
new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
  if (state.deleteSsmOnlyRoleError) {
    console.error(state.deleteSsmOnlyRoleError);
    return MESSAGES.deleteSsmOnlyRoleError.replace(
      "${ROLE_NAME}",
      NAMES.ssmOnlyRoleName,
    );
  } else {
```

```
        return MESSAGES.deletedSsmOnlyRole.replace(
            "${ROLE_NAME}",
            NAMES.ssmOnlyRoleName,
        );
    }
}),
];

/**
 * @param {string} policyName
 */
async function findPolicy(policyName) {
    const client = new IAMClient({});
    const paginatedPolicies = paginateListPolicies({ client }, {});
    for await (const page of paginatedPolicies) {
        const policy = page.Policies.find((p) => p.PolicyName === policyName);
        if (policy) {
            return policy;
        }
    }
}

/**
 * @param {string} groupName
 */
async function deleteAutoScalingGroup(groupName) {
    const client = new AutoScalingClient({});
    try {
        await client.send(
            new DeleteAutoScalingGroupCommand({
                AutoScalingGroupName: groupName,
            }),
        );
    } catch (err) {
        if (!(err instanceof Error)) {
            throw err;
        } else {
            console.log(err.name);
            throw err;
        }
    }
}

/**
```

```
* @param {string} groupName
*/
async function terminateGroupInstances(groupName) {
  const autoScalingClient = new AutoScalingClient({});
  const group = await findAutoScalingGroup(groupName);
  await autoScalingClient.send(
    new UpdateAutoScalingGroupCommand({
      AutoScalingGroupName: group.AutoScalingGroupName,
      MinSize: 0,
    }),
  );
  for (const i of group.Instances) {
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: i.InstanceId,
          ShouldDecrementDesiredCapacity: true,
        }),
      ),
    );
  }
}

async function findAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
  for await (const page of paginatedGroups) {
    const group = page.AutoScalingGroups.find(
      (g) => g.AutoScalingGroupName === groupName,
    );
    if (group) {
      return group;
    }
  }
  throw new Error(`Auto scaling group ${groupName} not found.`);
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for JavaScript .
 - [AttachLoadBalancerTargetGroups](#)
 - [CreateAutoScalingGroup](#)

- [CreateInstanceProfile](#)
- [CreateLaunchTemplate](#)
- [CreateListener](#)
- [CreateLoadBalancer](#)
- [CreateTargetGroup](#)
- [DeleteAutoScalingGroup](#)
- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplacelamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

Python

SDK pour Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Exécutez un scénario interactif à une invite de commande.

```
class Runner:
    def __init__(
        self, resource_path, recommendation, autoscaler, loadbalancer,
        param_helper
    ):
        self.resource_path = resource_path
        self.recommendation = recommendation
        self.autoscaler = autoscaler
        self.loadbalancer = loadbalancer
        self.param_helper = param_helper
        self.protocol = "HTTP"
        self.port = 80
        self.ssh_port = 22

    def deploy(self):
        recommendations_path = f"{self.resource_path}/recommendations.json"
        startup_script = f"{self.resource_path}/server_startup_script.sh"
        instance_policy = f"{self.resource_path}/instance_policy.json"

        print(
            "\nFor this demo, we'll use the AWS SDK for Python (Boto3) to create
            several AWS resources\n"
            "to set up a load-balanced web service endpoint and explore some ways
            to make it resilient\n"
            "against various kinds of failures.\n\n"
            "Some of the resources create by this demo are:\n"
        )
        print(
            "\t* A DynamoDB table that the web service depends on to provide
            book, movie, and song recommendations."
        )
        print(
            "\t* An EC2 launch template that defines EC2 instances that each
            contain a Python web server."
        )
        print(
            "\t* An EC2 Auto Scaling group that manages EC2 instances across
            several Availability Zones."
        )
        print(
            "\t* An Elastic Load Balancing (ELB) load balancer that targets the
            Auto Scaling group to distribute requests."
```

```
)
print("-" * 88)
q.ask("Press Enter when you're ready to start deploying resources.")

print(
    f"Creating and populating a DynamoDB table named
'{self.recommendation.table_name}'."
)
self.recommendation.create()
self.recommendation.populate(recommendations_path)
print("-" * 88)

print(
    f"Creating an EC2 launch template that runs '{startup_script}' when
an instance starts.\n"
    f"This script starts a Python web server defined in the `server.py`
script. The web server\n"
    f"listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.\n"
    f"For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
    f"run a web server, such as Apache, with least-privileged
credentials.\n"
)
print(
    f"The template also defines an IAM policy that each instance uses to
assume a role that grants\n"
    f"permissions to access the DynamoDB recommendation table and Systems
Manager parameters\n"
    f"that control the flow of the demo.\n"
)
self.autoscaler.create_template(startup_script, instance_policy)
print("-" * 88)

print(
    f"Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different\n"
    f"Availability Zone."
)
zones = self.autoscaler.create_group(3)
print("-" * 88)
print(
    "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
```

```
        "HTTP requests. You can see these instances in the console or
continue with the demo."
    )
    print("-" * 88)
    q.ask("Press Enter when you're ready to continue.")

    print(f"Creating variables that control the flow of the demo.\n")
    self.param_helper.reset()

    print(
        "\nCreating an Elastic Load Balancing target group and load balancer.
The target group\n"
        "defines how the load balancer connects to instances. The load
balancer provides a\n"
        "single endpoint where clients connect and dispatches requests to
instances in the group.\n"
    )
    vpc = self.autoscaler.get_default_vpc()
    subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
    target_group = self.loadbalancer.create_target_group(
        self.protocol, self.port, vpc["VpcId"]
    )
    self.loadbalancer.create_load_balancer(
        [subnet["SubnetId"] for subnet in subnets], target_group
    )
    self.autoscaler.attach_load_balancer_target_group(target_group)
    print(f"Verifying access to the load balancer endpoint...")
    lb_success = self.loadbalancer.verify_load_balancer_endpoint()
    if not lb_success:
        print(
            "Couldn't connect to the load balancer, verifying that the port
is open..."
        )
        current_ip_address = requests.get(
            "http://checkip.amazonaws.com"
        ).text.strip()
        sec_group, port_is_open = self.autoscaler.verify_inbound_port(
            vpc, self.port, current_ip_address
        )
        sec_group, ssh_port_is_open = self.autoscaler.verify_inbound_port(
            vpc, self.ssh_port, current_ip_address
        )
        if not port_is_open:
            print(
```

```

        "For this example to work, the default security group for
your default VPC must\n"
        "allows access from this computer. You can either add it
automatically from this\n"
        "example or add it yourself using the AWS Management Console.
\n"
    )
    if q.ask(
        f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
        f"inbound traffic on port {self.port} from your computer's IP
address of {current_ip_address}? (y/n) ",
        q.is_yesno,
    ):
        self.autoscaler.open_inbound_port(
            sec_group["GroupId"], self.port, current_ip_address
        )
    if not ssh_port_is_open:
        if q.ask(
            f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
            f"inbound SSH traffic on port {self.ssh_port} for debugging
from your computer's IP address of {current_ip_address}? (y/n) ",
            q.is_yesno,
        ):
            self.autoscaler.open_inbound_port(
                sec_group["GroupId"], self.ssh_port, current_ip_address
            )
    lb_success = self.loadbalancer.verify_load_balancer_endpoint()
    if lb_success:
        print("Your load balancer is ready. You can access it by browsing to:
\n")
        print(f"\thttp://{self.loadbalancer.endpoint()}\n")
    else:
        print(
            "Couldn't get a successful response from the load balancer
endpoint. Troubleshoot by\n"
            "manually verifying that your VPC and security group are
configured correctly and that\n"
            "you can successfully make a GET request to the load balancer
endpoint:\n"
        )
        print(f"\thttp://{self.loadbalancer.endpoint()}\n")
    print("-" * 88)

```

```
q.ask("Press Enter when you're ready to continue with the demo.")

def demo_choices(self):
    actions = [
        "Send a GET request to the load balancer endpoint.",
        "Check the health of load balancer targets.",
        "Go to the next part of the demo.",
    ]
    choice = 0
    while choice != 2:
        print("-" * 88)
        print(
            "\nSee the current state of the service by selecting one of the
following choices:\n"
        )
        choice = q.choose("\nWhich action would you like to take? ", actions)
        print("-" * 88)
        if choice == 0:
            print("Request:\n")
            print(f"GET http://{self.loadbalancer.endpoint()}")
            response = requests.get(f"http://{self.loadbalancer.endpoint()}")
            print("\nResponse:\n")
            print(f"{response.status_code}")
            if response.headers.get("content-type") == "application/json":
                pp(response.json())
        elif choice == 1:
            print("\nChecking the health of load balancer targets:\n")
            health = self.loadbalancer.check_target_health()
            for target in health:
                state = target["TargetHealth"]["State"]
                print(
                    f"\tTarget {target['Target']['Id']} on port
{target['Target']['Port']} is {state}"
                )
                if state != "healthy":
                    print(
                        f"\t\t{target['TargetHealth']['Reason']}:
{target['TargetHealth']['Description']}\n"
                    )
            print(
                f"\nNote that it can take a minute or two for the health
check to update\n"
                f"after changes are made.\n"
            )
    )
```

```
        elif choice == 2:
            print("\nOkay, let's move on.")
            print("-" * 88)

    def demo(self):
        ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

        print("\nResetting parameters to starting values for demo.\n")
        self.param_helper.reset()

        print(
            "\nThis part of the demonstration shows how to toggle different parts
of the system\n"
            "to create situations where the web service fails, and shows how
using a resilient\n"
            "architecture can keep the web service running in spite of these
failures."
        )
        print("-" * 88)

        print(
            "At the start, the load balancer endpoint returns recommendations and
reports that all targets are healthy."
        )
        self.demo_choices()

        print(
            f"The web service running on the EC2 instances gets recommendations
by querying a DynamoDB table.\n"
            f"The table name is contained in a Systems Manager parameter named
'{self.param_helper.table}'.\n"
            f"To simulate a failure of the recommendation service, let's set this
parameter to name a non-existent table.\n"
        )
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        print(
            "\nNow, sending a GET request to the load balancer endpoint returns a
failure code. But, the service reports as\n"
            "healthy to the load balancer because shallow health checks don't
check for failure of the recommendation service."
        )
        self.demo_choices()

        print(
```

```
        f"Instead of failing when the recommendation service fails, the web
service can return a static response.\n"
        f"While this is not a perfect solution, it presents the customer with
a somewhat better experience than failure.\n"
    )
    self.param_helper.put(self.param_helper.failure_response, "static")
    print(
        f"\nNow, sending a GET request to the load balancer endpoint returns
a static response.\n"
        f"The service still reports as healthy because health checks are
still shallow.\n"
    )
    self.demo_choices()

    print("Let's reinstate the recommendation service.\n")
    self.param_helper.put(self.param_helper.table,
self.recommendation.table_name)
    print(
        "\nLet's also substitute bad credentials for one of the instances in
the target group so that it can't\n"
        "access the DynamoDB recommendation table.\n"
    )
    self.autoscaler.create_instance_profile(
        ssm_only_policy,
        self.autoscaler.bad_creds_policy_name,
        self.autoscaler.bad_creds_role_name,
        self.autoscaler.bad_creds_profile_name,
        ["AmazonSSMManagedInstanceCore"],
    )
    instances = self.autoscaler.get_instances()
    bad_instance_id = instances[0]
    instance_profile = self.autoscaler.get_instance_profile(bad_instance_id)
    print(
        f"\nReplacing the profile for instance {bad_instance_id} with a
profile that contains\n"
        f"bad credentials...\n"
    )
    self.autoscaler.replace_instance_profile(
        bad_instance_id,
        self.autoscaler.bad_creds_profile_name,
        instance_profile["AssociationId"],
    )
    print(
```

```
        "Now, sending a GET request to the load balancer endpoint returns
either a recommendation or a static response,\n"
        "depending on which instance is selected by the load balancer.\n"
    )
    self.demo_choices()

    print(
        "\nLet's implement a deep health check. For this demo, a deep health
check tests whether\n"
        "the web service can access the DynamoDB table that it depends on for
recommendations. Note that\n"
        "the deep health check is only for ELB routing and not for Auto
Scaling instance health.\n"
        "This kind of deep health check is not recommended for Auto Scaling
instance health, because it\n"
        "risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.\n"
    )
    print(
        "By implementing deep health checks, the load balancer can detect
when one of the instances is failing\n"
        "and take that instance out of rotation.\n"
    )
    self.param_helper.put(self.param_helper.health_check, "deep")
    print(
        f"\nNow, checking target health indicates that the instance with bad
credentials ({bad_instance_id})\n"
        f"is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy \n"
        f"instance. Sending a GET request to the load balancer endpoint
always returns a recommendation, because\n"
        "the load balancer takes unhealthy instances out of its rotation.\n"
    )
    self.demo_choices()

    print(
        "\nBecause the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy\n"
        "instance is to terminate it and let the auto scaler start a new
instance to replace it.\n"
    )
    self.autoscaler.terminate_instance(bad_instance_id)
    print(
```

```
        "\nEven while the instance is terminating and the new instance is
starting, sending a GET\n"
        "request to the web service continues to get a successful
recommendation response because\n"
        "the load balancer routes requests to the healthy instances. After
the replacement instance\n"
        "starts and reports as healthy, it is included in the load balancing
rotation.\n"
        "\nNote that terminating and replacing an instance typically takes
several minutes, during which time you\n"
        "can see the changing health check status until the new instance is
running and healthy.\n"
    )
    self.demo_choices()

    print(
        "\nIf the recommendation service fails now, deep health checks mean
all instances report as unhealthy.\n"
    )
    self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
    print(
        "\nWhen all instances are unhealthy, the load balancer continues to
route requests even to\n"
        "unhealthy instances, allowing them to fail open and return a static
response rather than fail\n"
        "closed and report failure to the customer."
    )
    self.demo_choices()
    self.param_helper.reset()

    def destroy(self):
        print(
            "This concludes the demo of how to build and manage a resilient
service.\n"
            "To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources\n"
            "that were created for this demo."
        )
        if q.ask("Do you want to clean up all demo resources? (y/n) ",
q.is_yesno):
            self.loadbalancer.delete_load_balancer()
            self.loadbalancer.delete_target_group()
            self.autoscaler.delete_group()
            self.autoscaler.delete_key_pair()
```

```
        self.autoscaler.delete_template()
        self.autoscaler.delete_instance_profile(
            self.autoscaler.bad_creds_profile_name,
            self.autoscaler.bad_creds_role_name,
        )
        self.recommendation.destroy()
    else:
        print(
            "Okay, we'll leave the resources intact.\n"
            "Don't forget to delete them when you're done with them or you
            might incur unexpected charges."
        )

def main():
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--action",
        required=True,
        choices=["all", "deploy", "demo", "destroy"],
        help="The action to take for the demo. When 'all' is specified, resources
        are\n"
        "deployed, the demo is run, and resources are destroyed.",
    )
    parser.add_argument(
        "--resource_path",
        default="../../../workflows/resilient_service/resources",
        help="The path to resource files used by this example, such as IAM
        policies and\n"
        "instance scripts.",
    )
    args = parser.parse_args()

    print("-" * 88)
    print(
        "Welcome to the demonstration of How to Build and Manage a Resilient
        Service!"
    )
    print("-" * 88)

    prefix = "doc-example-resilience"
    recommendation = RecommendationService.from_client(
        "doc-example-recommendation-service"
    )
```

```

autoscaler = AutoScaler.from_client(prefix)
loadbalancer = LoadBalancer.from_client(prefix)
param_helper = ParameterHelper.from_client(recommendation.table_name)
runner = Runner(
    args.resource_path, recommendation, autoscaler, loadbalancer,
param_helper
)
actions = [args.action] if args.action != "all" else ["deploy", "demo",
"destroy"]
for action in actions:
    if action == "deploy":
        runner.deploy()
    elif action == "demo":
        runner.demo()
    elif action == "destroy":
        runner.destroy()

print("-" * 88)
print("Thanks for watching!")
print("-" * 88)

if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    main()

```

Créez une classe qui englobe les actions Auto Scaling et Amazon EC2.

```

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

    def __init__(
        self,
        resource_prefix,
        inst_type,
        ami_param,
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,

```

```

):
    """
    :param resource_prefix: The prefix for naming AWS resources that are
    created by this class.
    :param inst_type: The type of EC2 instance to create, such as t3.micro.
    :param ami_param: The Systems Manager parameter used to look up the AMI
    that is
        created.
    :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
    :param ec2_client: A Boto3 EC2 client.
    :param ssm_client: A Boto3 Systems Manager client.
    :param iam_client: A Boto3 IAM client.
    """
    self.inst_type = inst_type
    self.ami_param = ami_param
    self.autoscaling_client = autoscaling_client
    self.ec2_client = ec2_client
    self.ssm_client = ssm_client
    self.iam_client = iam_client
    self.launch_template_name = f"{resource_prefix}-template"
    self.group_name = f"{resource_prefix}-group"
    self.instance_policy_name = f"{resource_prefix}-pol"
    self.instance_role_name = f"{resource_prefix}-role"
    self.instance_profile_name = f"{resource_prefix}-prof"
    self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
    self.bad_creds_role_name = f"{resource_prefix}-bc-role"
    self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"
    self.key_pair_name = f"{resource_prefix}-key-pair"

    @classmethod
    def from_client(cls, resource_prefix):
        """
        Creates this class from Boto3 clients.

        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        """
        as_client = boto3.client("autoscaling")
        ec2_client = boto3.client("ec2")
        ssm_client = boto3.client("ssm")
        iam_client = boto3.client("iam")
        return cls(
            resource_prefix,

```

```

        "t3.micro",
        "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
        as_client,
        ec2_client,
        ssm_client,
        iam_client,
    )

    def create_instance_profile(
        self, policy_file, policy_name, role_name, profile_name,
        aws_managed_policies=()
    ):
        """
        Creates a policy, role, and profile that is associated with instances
        created by
        this class. An instance's associated profile defines a role that is
        assumed by the
        instance. The role has attached policies that specify the AWS permissions
        granted to
        clients that run on the instance.

        :param policy_file: The name of a JSON file that contains the policy
        definition to
            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
        attached to
            the role, such as
        AmazonSSMManagedInstanceCore to grant
            use of Systems Manager to send commands to
        the instance.

        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        ],

```

```

    }
    with open(policy_file) as file:
        instance_policy_doc = file.read()

    policy_arn = None
    try:
        pol_response = self.iam_client.create_policy(
            PolicyName=policy_name, PolicyDocument=instance_policy_doc
        )
        policy_arn = pol_response["Policy"]["Arn"]
        log.info("Created policy with ARN %s.", policy_arn)
    except ClientError as err:
        if err.response["Error"]["Code"] == "EntityAlreadyExists":
            log.info("Policy %s already exists, nothing to do.", policy_name)
            list_pol_response = self.iam_client.list_policies(Scope="Local")
            for pol in list_pol_response["Policies"]:
                if pol["PolicyName"] == policy_name:
                    policy_arn = pol["Arn"]
                    break
            if policy_arn is None:
                raise AutoScalerError(f"Couldn't create policy {policy_name}:
{err}")

        try:
            self.iam_client.create_role(
                RoleName=role_name,
                AssumeRolePolicyDocument=json.dumps(assume_role_doc)
            )
            self.iam_client.attach_role_policy(RoleName=role_name,
                PolicyArn=policy_arn)
            for aws_policy in aws_managed_policies:
                self.iam_client.attach_role_policy(
                    RoleName=role_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
                )
            log.info("Created role %s and attached policy %s.", role_name,
                policy_arn)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                log.info("Role %s already exists, nothing to do.", role_name)
            else:
                raise AutoScalerError(f"Couldn't create role {role_name}: {err}")

        try:

```

```
        profile_response = self.iam_client.create_instance_profile(
            InstanceProfileName=profile_name
        )
        waiter = self.iam_client.get_waiter("instance_profile_exists")
        waiter.wait(InstanceProfileName=profile_name)
        time.sleep(10) # wait a little longer
        profile_arn = profile_response["InstanceProfile"]["Arn"]
        self.iam_client.add_role_to_instance_profile(
            InstanceProfileName=profile_name, RoleName=role_name
        )
        log.info("Created profile %s and added role %s.", profile_name,
role_name)
    except ClientError as err:
        if err.response["Error"]["Code"] == "EntityAlreadyExists":
            prof_response = self.iam_client.get_instance_profile(
                InstanceProfileName=profile_name
            )
            profile_arn = prof_response["InstanceProfile"]["Arn"]
            log.info(
                "Instance profile %s already exists, nothing to do.",
profile_name
            )
        else:
            raise AutoScalerError(
                f"Couldn't create profile {profile_name} and attach it to
role\n"
                f"{role_name}: {err}"
            )
    return profile_arn

def get_instance_profile(self, instance_id):
    """
    Gets data about the profile associated with an instance.

    :param instance_id: The ID of the instance to look up.
    :return: The profile data.
    """
    try:
        response =
self.ec2_client.describe_iam_instance_profile_associations(
            Filters=[{"Name": "instance-id", "Values": [instance_id]}]
        )
    except ClientError as err:
```

```
        raise AutoScalerError(
            f"Couldn't get instance profile association for instance
{instance_id}: {err}"
        )
    else:
        return response["IamInstanceProfileAssociations"][0]

def replace_instance_profile(
    self, instance_id, new_instance_profile_name, profile_association_id
):
    """
    Replaces the profile associated with a running instance. After the
    profile is
    replaced, the instance is rebooted to ensure that it uses the new
    profile. When
    the instance is ready, Systems Manager is used to restart the Python web
    server.

    :param instance_id: The ID of the instance to update.
    :param new_instance_profile_name: The name of the new profile to
    associate with
                                the specified instance.
    :param profile_association_id: The ID of the existing profile association
    for the
                                instance.
    """
    try:
        self.ec2_client.replace_iam_instance_profile_association(
            IamInstanceProfile={"Name": new_instance_profile_name},
            AssociationId=profile_association_id,
        )
        log.info(
            "Replaced instance profile for association %s with profile %s.",
            profile_association_id,
            new_instance_profile_name,
        )
        time.sleep(5)
        inst_ready = False
        tries = 0
        while not inst_ready:
            if tries % 6 == 0:
                self.ec2_client.reboot_instances(InstanceIds=[instance_id])
                log.info(
```

```
        "Rebooting instance %s and waiting for it to to be
ready.",
        instance_id,
    )
    tries += 1
    time.sleep(10)
    response = self.ssm_client.describe_instance_information()
    for info in response["InstanceInformationList"]:
        if info["InstanceId"] == instance_id:
            inst_ready = True
    self.ssm_client.send_command(
        InstanceIds=[instance_id],
        DocumentName="AWS-RunShellScript",
        Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
    )
    log.info("Restarted the Python web server on instance %s.",
instance_id)
    except ClientError as err:
        raise AutoScalerError(
            f"Couldn't replace instance profile for association
{profile_association_id}: {err}"
        )

def delete_instance_profile(self, profile_name, role_name):
    """
    Detaches a role from an instance profile, detaches policies from the
role,
and deletes all the resources.

:param profile_name: The name of the profile to delete.
:param role_name: The name of the role to delete.
    """
    try:
        self.iam_client.remove_role_from_instance_profile(
            InstanceProfileName=profile_name, RoleName=role_name
        )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
    log.info("Deleted instance profile %s.", profile_name)
    attached_policies = self.iam_client.list_attached_role_policies(
        RoleName=role_name
    )
    for pol in attached_policies["AttachedPolicies"]:
```

```
        self.iam_client.detach_role_policy(
            RoleName=role_name, PolicyArn=pol["PolicyArn"]
        )
        if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
            self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
            log.info("Detached and deleted policy %s.", pol["PolicyName"])
        self.iam_client.delete_role(RoleName=role_name)
        log.info("Deleted role %s.", role_name)
    except ClientError as err:
        if err.response["Error"]["Code"] == "NoSuchEntity":
            log.info(
                "Instance profile %s doesn't exist, nothing to do.",
profile_name
            )
        else:
            raise AutoScalerError(
                f"Couldn't delete instance profile {profile_name} or detach "
                f"policies and delete role {role_name}: {err}"
            )

def create_key_pair(self, key_pair_name):
    """
    Creates a new key pair.

    :param key_pair_name: The name of the key pair to create.
    :return: The newly created key pair.
    """
    try:
        response = self.ec2_client.create_key_pair(KeyName=key_pair_name)
        with open(f"{key_pair_name}.pem", "w") as file:
            file.write(response["KeyMaterial"])
        chmod(f"{key_pair_name}.pem", 0o600)
        log.info("Created key pair %s.", key_pair_name)
    except ClientError as err:
        raise AutoScalerError(f"Couldn't create key pair {key_pair_name}:
{err}")

def delete_key_pair(self):
    """
    Deletes a key pair.

    :param key_pair_name: The name of the key pair to delete.
```

```

    """
    try:
        self.ec2_client.delete_key_pair(KeyName=self.key_pair_name)
        remove(f"{self.key_pair_name}.pem")
        log.info("Deleted key pair %s.", self.key_pair_name)
    except ClientError as err:
        raise AutoScalerError(
            f"Couldn't delete key pair {self.key_pair_name}: {err}"
        )
    except FileNotFoundError:
        log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
    except PermissionError:
        log.info(
            "Inadequate permissions to delete key pair %s.",
self.key_pair_name
        )
    except Exception as err:
        raise AutoScalerError(
            f"Couldn't delete key pair {self.key_pair_name}: {err}"
        )

    def create_template(self, server_startup_script_file, instance_policy_file):
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling. The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts
a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
            when an instance starts.
        :param instance_policy_file: The path to a file that defines a
permissions policy
            to create and attach to the instance
profile.
        :return: Information about the newly created template.
        """
        template = {}
        try:

```

```
self.create_key_pair(self.key_pair_name)
self.create_instance_profile(
    instance_policy_file,
    self.instance_policy_name,
    self.instance_role_name,
    self.instance_profile_name,
)
with open(server_startup_script_file) as file:
    start_server_script = file.read()
ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
ami_id = ami_latest["Parameter"]["Value"]
lt_response = self.ec2_client.create_launch_template(
    LaunchTemplateName=self.launch_template_name,
    LaunchTemplateData={
        "InstanceType": self.inst_type,
        "ImageId": ami_id,
        "IamInstanceProfile": {"Name": self.instance_profile_name},
        "UserData": base64.b64encode(
            start_server_script.encode(encoding="utf-8")
        ).decode(encoding="utf-8"),
        "KeyName": self.key_pair_name,
    },
)
template = lt_response["LaunchTemplate"]
log.info(
    "Created launch template %s for AMI %s on %s.",
    self.launch_template_name,
    ami_id,
    self.inst_type,
)
except ClientError as err:
    if (
        err.response["Error"]["Code"]
        == "InvalidLaunchTemplateName.AlreadyExistsException"
    ):
        log.info(
            "Launch template %s already exists, nothing to do.",
            self.launch_template_name,
        )
    else:
        raise AutoScalerError(
            f"Couldn't create launch template
{self.launch_template_name}: {err}."
        )
```

```
    return template

def delete_template(self):
    """
    Deletes a launch template.
    """
    try:
        self.ec2_client.delete_launch_template(
            LaunchTemplateName=self.launch_template_name
        )
        self.delete_instance_profile(
            self.instance_profile_name, self.instance_role_name
        )
        log.info("Launch template %s deleted.", self.launch_template_name)
    except ClientError as err:
        if (
            err.response["Error"]["Code"]
            == "InvalidLaunchTemplateName.NotFoundException"
        ):
            log.info(
                "Launch template %s does not exist, nothing to do.",
                self.launch_template_name,
            )
        else:
            raise AutoScalerError(
                f"Couldn't delete launch template
{self.launch_template_name}: {err}."
            )

def get_availability_zones(self):
    """
    Gets a list of Availability Zones in the AWS Region of the Amazon EC2
    client.

    :return: The list of Availability Zones for the client Region.
    """
    try:
        response = self.ec2_client.describe_availability_zones()
        zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
    except ClientError as err:
        raise AutoScalerError(f"Couldn't get availability zones: {err}.")
    else:
```

```
        return zones

def create_group(self, group_size):
    """
    Creates an EC2 Auto Scaling group with the specified size.

    :param group_size: The number of instances to set for the minimum and
maximum in
                        the group.
    :return: The list of Availability Zones specified for the group.
    """
    zones = []
    try:
        zones = self.get_availability_zones()
        self.autoscaling_client.create_auto_scaling_group(
            AutoScalingGroupName=self.group_name,
            AvailabilityZones=zones,
            LaunchTemplate={
                "LaunchTemplateName": self.launch_template_name,
                "Version": "$Default",
            },
            MinSize=group_size,
            MaxSize=group_size,
        )
        log.info(
            "Created EC2 Auto Scaling group %s with availability zones %s.",
            self.launch_template_name,
            zones,
        )
    except ClientError as err:
        if err.response["Error"]["Code"] == "AlreadyExists":
            log.info(
                "EC2 Auto Scaling group %s already exists, nothing to do.",
                self.group_name,
            )
        else:
            raise AutoScalerError(
                f"Couldn't create EC2 Auto Scaling group {self.group_name}:
{err}")
    return zones
```

```
def get_instances(self):
    """
    Gets data about the instances in the EC2 Auto Scaling group.

    :return: Data about the instances.
    """
    try:
        as_response = self.autoscaling_client.describe_auto_scaling_groups(
            AutoScalingGroupNames=[self.group_name]
        )
        instance_ids = [
            i["InstanceId"]
            for i in as_response["AutoScalingGroups"][0]["Instances"]
        ]
    except ClientError as err:
        raise AutoScalerError(
            f"Couldn't get instances for Auto Scaling group
{self.group_name}: {err}"
        )
    else:
        return instance_ids

def terminate_instance(self, instance_id):
    """
    Terminates and instances in an EC2 Auto Scaling group. After an instance
is
    terminated, it can no longer be accessed.

    :param instance_id: The ID of the instance to terminate.
    """
    try:
        self.autoscaling_client.terminate_instance_in_auto_scaling_group(
            InstanceId=instance_id, ShouldDecrementDesiredCapacity=False
        )
        log.info("Terminated instance %s.", instance_id)
    except ClientError as err:
        raise AutoScalerError(f"Couldn't terminate instance {instance_id}:
{err}")

def attach_load_balancer_target_group(self, lb_target_group):
    """
    Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
```

```
The target group specifies how the load balancer forward requests to the
instances
in the group.

:param lb_target_group: Data about the ELB target group to attach.
"""
try:
    self.autoscaling_client.attach_load_balancer_target_groups(
        AutoScalingGroupName=self.group_name,
        TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
    )
    log.info(
        "Attached load balancer target group %s to auto scaling group
%s.",
        lb_target_group["TargetGroupName"],
        self.group_name,
    )
except ClientError as err:
    raise AutoScalerError(
        f"Couldn't attach load balancer target group
{lb_target_group['TargetGroupName']}\n"
        f"to auto scaling group {self.group_name}"
    )

def _try_terminate_instance(self, inst_id):
    stopping = False
    log.info(f"Stopping {inst_id}.")
    while not stopping:
        try:
            self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=inst_id, ShouldDecrementDesiredCapacity=True
            )
            stopping = True
        except ClientError as err:
            if err.response["Error"]["Code"] == "ScalingActivityInProgress":
                log.info("Scaling activity in progress for %s. Waiting...",
inst_id)
                time.sleep(10)
            else:
                raise AutoScalerError(f"Couldn't stop instance {inst_id}:
{err}.")

def _try_delete_group(self):
```

```
    """
    Tries to delete the EC2 Auto Scaling group. If the group is in use or in
    progress,
    the function waits and retries until the group is successfully deleted.
    """
    stopped = False
    while not stopped:
        try:
            self.autoscaling_client.delete_auto_scaling_group(
                AutoScalingGroupName=self.group_name
            )
            stopped = True
            log.info("Deleted EC2 Auto Scaling group %s.", self.group_name)
        except ClientError as err:
            if (
                err.response["Error"]["Code"] == "ResourceInUse"
                or err.response["Error"]["Code"] ==
                "ScalingActivityInProgress"
            ):
                log.info(
                    "Some instances are still running. Waiting for them to
                    stop..."
                )
                time.sleep(10)
            else:
                raise AutoScalerError(
                    f"Couldn't delete group {self.group_name}: {err}."
                )

    def delete_group(self):
        """
        Terminates all instances in the group, deletes the EC2 Auto Scaling
        group.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[self.group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=self.group_name, MinSize=0
                )
```

```
        instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
        for inst_id in instance_ids:
            self._try_terminate_instance(inst_id)
            self._try_delete_group()
        else:
            log.info("No groups found named %s, nothing to do.",
self.group_name)
        except ClientError as err:
            raise AutoScalerError(f"Couldn't delete group {self.group_name}:
{err}.")

def get_default_vpc(self):
    """
    Gets the default VPC for the account.

    :return: Data about the default VPC.
    """
    try:
        response = self.ec2_client.describe_vpcs(
            Filters=[{"Name": "is-default", "Values": ["true"]}])
    except ClientError as err:
        raise AutoScalerError(f"Couldn't get default VPC: {err}")
    else:
        return response["Vpcs"][0]

def verify_inbound_port(self, vpc, port, ip_address):
    """
    Verify the default security group of the specified VPC allows ingress
from this
    computer. This can be done by allowing ingress from this computer's IP
address. In some situations, such as connecting from a corporate network,
you
    must instead specify a prefix list ID. You can also temporarily open the
port to
    any IP address while running this example. If you do, be sure to remove
public
    access when you're done.

    :param vpc: The VPC used by this example.
    :param port: The port to verify.
```

```

:param ip_address: This computer's IP address.
:return: The default security group of the specific VPC, and a value that
indicates
        whether the specified port is open.
"""
try:
    response = self.ec2_client.describe_security_groups(
        Filters=[
            {"Name": "group-name", "Values": ["default"]},
            {"Name": "vpc-id", "Values": [vpc["VpcId"]]},
        ]
    )
    sec_group = response["SecurityGroups"][0]
    port_is_open = False
    log.info("Found default security group %s.", sec_group["GroupId"])
    for ip_perm in sec_group["IpPermissions"]:
        if ip_perm.get("FromPort", 0) == port:
            log.info("Found inbound rule: %s", ip_perm)
            for ip_range in ip_perm["IpRanges"]:
                cidr = ip_range.get("CidrIp", "")
                if cidr.startswith(ip_address) or cidr == "0.0.0.0/0":
                    port_is_open = True
            if ip_perm["PrefixListIds"]:
                port_is_open = True
            if not port_is_open:
                log.info(
                    "The inbound rule does not appear to be open to
either this computer's IP\n"
                    "address of %s, to all IP addresses (0.0.0.0/0), or
to a prefix list ID.",
                    ip_address,
                )
            else:
                break
    except ClientError as err:
        raise AutoScalerError(
            f"Couldn't verify inbound rule for port {port} for VPC
{vpc['VpcId']}: {err}"
        )
    else:
        return sec_group, port_is_open

def open_inbound_port(self, sec_group_id, port, ip_address):

```

```
"""
Add an ingress rule to the specified security group that allows access on
the
specified port from the specified IP address.

:param sec_group_id: The ID of the security group to modify.
:param port: The port to open.
:param ip_address: The IP address that is granted access.
"""
try:
    self.ec2_client.authorize_security_group_ingress(
        GroupId=sec_group_id,
        CidrIp=f"{ip_address}/32",
        FromPort=port,
        ToPort=port,
        IpProtocol="tcp",
    )
    log.info(
        "Authorized ingress to %s on port %s from %s.",
        sec_group_id,
        port,
        ip_address,
    )
except ClientError as err:
    raise AutoScalerError(
        f"Couldn't authorize ingress to {sec_group_id} on port {port}
from {ip_address}: {err}"
    )

def get_subnets(self, vpc_id, zones):
    """
    Gets the default subnets in a VPC for a specified list of Availability
    Zones.

    :param vpc_id: The ID of the VPC to look up.
    :param zones: The list of Availability Zones to look up.
    :return: The list of subnets found.
    """
    try:
        response = self.ec2_client.describe_subnets(
            Filters=[
                {"Name": "vpc-id", "Values": [vpc_id]},
                {"Name": "availability-zone", "Values": zones},
            ]
        )
```

```

        {"Name": "default-for-az", "Values": ["true"]},
    ]
)
subnets = response["Subnets"]
log.info("Found %s subnets for the specified zones.", len(subnets))
except ClientError as err:
    raise AutoScalerError(f"Couldn't get subnets: {err}")
else:
    return subnets

```

Créez une classe qui englobe les actions Elastic Load Balancing.

```

class LoadBalancer:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, target_group_name, load_balancer_name, elb_client):
        """
        :param target_group_name: The name of the target group associated with
        the load balancer.
        :param load_balancer_name: The name of the load balancer.
        :param elb_client: A Boto3 Elastic Load Balancing client.
        """
        self.target_group_name = target_group_name
        self.load_balancer_name = load_balancer_name
        self.elb_client = elb_client
        self._endpoint = None

    @classmethod
    def from_client(cls, resource_prefix):
        """
        Creates this class from a Boto3 client.

        :param resource_prefix: The prefix to give to AWS resources created by
        this class.
        """
        elb_client = boto3.client("elbv2")
        return cls(f"{resource_prefix}-tg", f"{resource_prefix}-lb", elb_client)

```

```
def endpoint(self):
    """
    Gets the HTTP endpoint of the load balancer.

    :return: The endpoint.
    """
    if self._endpoint is None:
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[self.load_balancer_name]
            )
            self._endpoint = response["LoadBalancers"][0]["DNSName"]
        except ClientError as err:
            raise LoadBalancerError(
                f"Couldn't get the endpoint for load balancer
{self.load_balancer_name}: {err}")
        return self._endpoint

def create_target_group(self, protocol, port, vpc_id):
    """
    Creates an Elastic Load Balancing target group. The target group
specifies how
    the load balancer forward requests to instances in the group and how
instance
    health is checked.

    To speed up this demo, the health check is configured with shortened
times and
    lower thresholds. In production, you might want to decrease the
sensitivity of
    your health checks to avoid unwanted failures.

    :param protocol: The protocol to use to forward requests, such as 'HTTP'.
    :param port: The port to use to forward requests, such as 80.
    :param vpc_id: The ID of the VPC in which the load balancer exists.
    :return: Data about the newly created target group.
    """
    try:
        response = self.elb_client.create_target_group(
            Name=self.target_group_name,
            Protocol=protocol,
            Port=port,
```

```

        HealthCheckPath="/healthcheck",
        HealthCheckIntervalSeconds=10,
        HealthCheckTimeoutSeconds=5,
        HealthyThresholdCount=2,
        UnhealthyThresholdCount=2,
        VpcId=vpc_id,
    )
    target_group = response["TargetGroups"][0]
    log.info("Created load balancing target group %s.",
self.target_group_name)
    except ClientError as err:
        raise LoadBalancerError(
            f"Couldn't create load balancing target group
{self.target_group_name}: {err}")
    )
    else:
        return target_group

def delete_target_group(self):
    """
    Deletes the target group.
    """
    done = False
    while not done:
        try:
            response = self.elb_client.describe_target_groups(
                Names=[self.target_group_name]
            )
            tg_arn = response["TargetGroups"][0]["TargetGroupArn"]
            self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
            log.info(
                "Deleted load balancing target group %s.",
self.target_group_name
            )
            done = True
        except ClientError as err:
            if err.response["Error"]["Code"] == "TargetGroupNotFound":
                log.info(
                    "Load balancer target group %s not found, nothing to
do.",
                    self.target_group_name,
                )
            done = True

```

```
        elif err.response["Error"]["Code"] == "ResourceInUse":
            log.info(
                "Target group not yet released from load balancer,
waiting..."
            )
            time.sleep(10)
        else:
            raise LoadBalancerError(
                f"Couldn't delete load balancing target group
{self.target_group_name}: {err}"
            )

    def create_load_balancer(self, subnet_ids, target_group):
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
subnets
and forwards requests to the specified target group.

:param subnet_ids: A list of subnets to associate with the load balancer.
:param target_group: An existing target group that is added as a listener
to the
                    load balancer.
:return: Data about the newly created load balancer.
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=self.load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info("Created load balancer %s.", self.load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info("Waiting for load balancer to be available...")
            waiter.wait(Names=[self.load_balancer_name])
            log.info("Load balancer is available!")
            self.elb_client.create_listener(
                LoadBalancerArn=load_balancer["LoadBalancerArn"],
                Protocol=target_group["Protocol"],
                Port=target_group["Port"],
                DefaultActions=[
                    {
                        "Type": "forward",
                        "TargetGroupArn": target_group["TargetGroupArn"],
                    }
                ]
            )
```

```

        ],
    )
    log.info(
        "Created listener to forward traffic from load balancer %s to
target group %s.",
        self.load_balancer_name,
        target_group["TargetGroupName"],
    )
except ClientError as err:
    raise LoadBalancerError(
        f"Failed to create load balancer {self.load_balancer_name}"
        f"and add a listener for target group
{target_group['TargetGroupName']}: {err}"
    )
else:
    self._endpoint = load_balancer["DNSName"]
    return load_balancer

def delete_load_balancer(self):
    """
    Deletes a load balancer.
    """
    try:
        response = self.elb_client.describe_load_balancers(
            Names=[self.load_balancer_name]
        )
        lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
        self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
        log.info("Deleted load balancer %s.", self.load_balancer_name)
        waiter = self.elb_client.get_waiter("load_balancers_deleted")
        log.info("Waiting for load balancer to be deleted...")
        waiter.wait(Names=[self.load_balancer_name])
    except ClientError as err:
        if err.response["Error"]["Code"] == "LoadBalancerNotFound":
            log.info(
                "Load balancer %s does not exist, nothing to do.",
                self.load_balancer_name,
            )
        else:
            raise LoadBalancerError(
                f"Couldn't delete load balancer {self.load_balancer_name}:
{err}"
            )

```

```
def verify_load_balancer_endpoint(self):
    """
    Verify this computer can successfully send a GET request to the load
    balancer endpoint.
    """
    success = False
    retries = 3
    while not success and retries > 0:
        try:
            lb_response = requests.get(f"http://{self.endpoint()}")
            log.info(
                "Got response %s from load balancer endpoint.",
                lb_response.status_code,
            )
            if lb_response.status_code == 200:
                success = True
            else:
                retries = 0
        except requests.exceptions.ConnectionError:
            log.info(
                "Got connection error from load balancer endpoint,
retrying..."
            )
            retries -= 1
            time.sleep(10)
    return success

def check_target_health(self):
    """
    Checks the health of the instances in the target group.

    :return: The health status of the target group.
    """
    try:
        tg_response = self.elb_client.describe_target_groups(
            Names=[self.target_group_name]
        )
        health_response = self.elb_client.describe_target_health(
            TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
        )
    except ClientError as err:
        raise LoadBalancerError(
```

```

        f"Couldn't check health of {self.target_group_name} targets:
{err}"
    )
    else:
        return health_response["TargetHealthDescriptions"]

```

Créez une classe qui utilise DynamoDB pour simuler un service de recommandation.

```

class RecommendationService:
    """
    Encapsulates a DynamoDB table to use as a service that recommends books,
    movies,
    and songs.
    """

    def __init__(self, table_name, dynamodb_client):
        """
        :param table_name: The name of the DynamoDB recommendations table.
        :param dynamodb_client: A Boto3 DynamoDB client.
        """
        self.table_name = table_name
        self.dynamodb_client = dynamodb_client

    @classmethod
    def from_client(cls, table_name):
        """
        Creates this class from a Boto3 client.

        :param table_name: The name of the DynamoDB recommendations table.
        """
        ddb_client = boto3.client("dynamodb")
        return cls(table_name, ddb_client)

    def create(self):
        """
        Creates a DynamoDB table to use a recommendation service. The table has a
        hash key named 'MediaType' that defines the type of media recommended,
        such as

```

```

    Book or Movie, and a range key named 'ItemId' that, combined with the
    MediaType,
    forms a unique identifier for the recommended item.

:return: Data about the newly created table.
"""
try:
    response = self.dynamodb_client.create_table(
        TableName=self.table_name,
        AttributeDefinitions=[
            {"AttributeName": "MediaType", "AttributeType": "S"},
            {"AttributeName": "ItemId", "AttributeType": "N"},
        ],
        KeySchema=[
            {"AttributeName": "MediaType", "KeyType": "HASH"},
            {"AttributeName": "ItemId", "KeyType": "RANGE"},
        ],
        ProvisionedThroughput={"ReadCapacityUnits": 5,
"WriteCapacityUnits": 5},
    )
    log.info("Creating table %s...", self.table_name)
    waiter = self.dynamodb_client.get_waiter("table_exists")
    waiter.wait(TableName=self.table_name)
    log.info("Table %s created.", self.table_name)
except ClientError as err:
    if err.response["Error"]["Code"] == "ResourceInUseException":
        log.info("Table %s exists, nothing to be do.", self.table_name)
    else:
        raise RecommendationServiceError(
            self.table_name, f"ClientError when creating table: {err}."
        )
else:
    return response

def populate(self, data_file):
    """
    Populates the recommendations table from a JSON file.

:param data_file: The path to the data file.
    """
    try:
        with open(data_file) as data:
            items = json.load(data)
            batch = [{"PutRequest": {"Item": item}} for item in items]

```

```

        self.dynamodb_client.batch_write_item(RequestItems={self.table_name:
batch})
        log.info(
            "Populated table %s with items from %s.", self.table_name,
data_file
        )
    except ClientError as err:
        raise RecommendationServiceError(
            self.table_name, f"Couldn't populate table from {data_file}:
{err}"
        )

    def destroy(self):
        """
        Deletes the recommendations table.
        """
        try:
            self.dynamodb_client.delete_table(TableName=self.table_name)
            log.info("Deleting table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_not_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s deleted.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                log.info("Table %s does not exist, nothing to do.",
self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when deleting table: {err}."
                )

```

Créez une classe qui englobe les actions de Systems Manager.

```

class ParameterHelper:
    """
    Encapsulates Systems Manager parameters. This example uses these parameters
to drive
the demonstration of resilient architecture, such as failure of a dependency
or
how the service responds to a health check.

```

```
"""

table = "doc-example-resilient-architecture-table"
failure_response = "doc-example-resilient-architecture-failure-response"
health_check = "doc-example-resilient-architecture-health-check"

def __init__(self, table_name, ssm_client):
    """
    :param table_name: The name of the DynamoDB table that is used as a
    recommendation
                        service.
    :param ssm_client: A Boto3 Systems Manager client.
    """
    self.ssm_client = ssm_client
    self.table_name = table_name

    @classmethod
    def from_client(cls, table_name):
        ssm_client = boto3.client("ssm")
        return cls(table_name, ssm_client)

    def reset(self):
        """
        Resets the Systems Manager parameters to starting values for the demo.
        These are the name of the DynamoDB recommendation table, no response when
a
        dependency fails, and shallow health checks.
        """
        self.put(self.table, self.table_name)
        self.put(self.failure_response, "none")
        self.put(self.health_check, "shallow")

    def put(self, name, value):
        """
        Sets the value of a named Systems Manager parameter.

        :param name: The name of the parameter.
        :param value: The new value of the parameter.
        """
        try:
            self.ssm_client.put_parameter(
                Name=name, Value=value, Overwrite=True, Type="String"
            )
            log.info("Setting demo parameter %s to '%s'.", name, value)
```

```
except ClientError as err:
    raise ParameterHelperError(
        f"Couldn't set parameter {name} to {value}: {err}"
    )
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans AWS SDK for Python (Boto3) API Reference.
 - [AttachLoadBalancerTargetGroups](#)
 - [CreateAutoScalingGroup](#)
 - [CreateInstanceProfile](#)
 - [CreateLaunchTemplate](#)
 - [CreateListener](#)
 - [CreateLoadBalancer](#)
 - [CreateTargetGroup](#)
 - [DeleteAutoScalingGroup](#)
 - [DeleteInstanceProfile](#)
 - [DeleteLaunchTemplate](#)
 - [DeleteLoadBalancer](#)
 - [DeleteTargetGroup](#)
 - [DescribeAutoScalingGroups](#)
 - [DescribeAvailabilityZones](#)
 - [DescribeIamInstanceProfileAssociations](#)
 - [DescribeInstances](#)
 - [DescribeLoadBalancers](#)
 - [DescribeSubnets](#)
 - [DescribeTargetGroups](#)
 - [DescribeTargetHealth](#)
 - [DescribeVpcs](#)
 - [RebootInstances](#)
 - [ReplacelamInstanceProfileAssociation](#)

- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Commencez à utiliser les instances Amazon EC2 à l'aide d'un SDK AWS

Les exemples de code suivants montrent comment :

- Créez une paire de clés et un groupe de sécurité.
- Sélectionnez une Amazon Machine Image (AMI) et un type d'instance compatible, puis créez une instance.
- Arrêtez l'instance, puis redémarrez-la.
- Associez une adresse IP Elastic à votre instance
- Connectez-vous à votre instance avec SSH, puis nettoyez les ressources.

.NET

AWS SDK for .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Exécutez un scénario à une invite de commande.

```
/// <summary>
/// Show Amazon Elastic Compute Cloud (Amazon EC2) Basics actions.
/// </summary>
public class EC2Basics
{
    /// <summary>
    /// Perform the actions defined for the Amazon EC2 Basics scenario.
    /// </summary>
```

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

    // Now the client is available for injection.
    var ec2Client = host.Services.GetRequiredService<IAmazonEC2>();
    var ec2Methods = new EC2Wrapper(ec2Client);

    var ssmClient =
host.Services.GetRequiredService<IAmazonSimpleSystemsManagement>();
    var ssmMethods = new SsmWrapper(ssmClient);
    var uiMethods = new UiMethods();

    var uniqueName = Guid.NewGuid().ToString();
    var keyPairName = "mvp-example-key-pair" + uniqueName;
    var groupName = "ec2-scenario-group" + uniqueName;
    var groupDescription = "A security group created for the EC2 Basics
scenario.";

    // Start the scenario.
    uiMethods.DisplayOverview();
    uiMethods.PressEnter();

    // Create the key pair.
    uiMethods.DisplayTitle("Create RSA key pair");
    Console.WriteLine("Let's create an RSA key pair that you can use to ");
    Console.WriteLine("securely connect to your EC2 instance.");
    var keyPair = await ec2Methods.CreateKeyPair(keyPairName);

    // Save key pair information to a temporary file.
    var tempFileName = ec2Methods.SaveKeyPair(keyPair);
```

```
    Console.WriteLine($"Created the key pair: {keyPair.KeyName} and saved it
to: {tempFileName}");
    string? answer;
    do
    {
        Console.Write("Would you like to list your existing key pairs? ");
        answer = Console.ReadLine();
    } while (answer!.ToLower() != "y" && answer.ToLower() != "n");

    if (answer == "y")
    {
        // List existing key pairs.
        uiMethods.DisplayTitle("Existing key pairs");

        // Passing an empty string to the DescribeKeyPairs method will return
        // a list of all existing key pairs.
        var keyPairs = await ec2Methods.DescribeKeyPairs("");
        keyPairs.ForEach(kp =>
        {
            Console.WriteLine($"{kp.KeyName} created at: {kp.CreateTime}
Fingerprint: {kp.KeyFingerprint}");
        });
        uiMethods.PressEnter();

        // Create the security group.
        Console.WriteLine("Let's create a security group to manage access to your
instance.");
        var secGroupId = await ec2Methods.CreateSecurityGroup(groupName,
groupDescription);
        Console.WriteLine("Let's add rules to allow all HTTP and HTTPS inbound
traffic and to allow SSH only from your current IP address.");

        uiMethods.DisplayTitle("Security group information");
        var secGroups = await ec2Methods.DescribeSecurityGroups(secGroupId);

        Console.WriteLine($"Created security group {groupName} in your default
VPC.");
        secGroups.ForEach(group =>
        {
            ec2Methods.DisplaySecurityGroupInfoAsync(group);
        });
        uiMethods.PressEnter();
    }
}
```

```
    Console.WriteLine("Now we'll authorize the security group we just created
so that it can");
    Console.WriteLine("access the EC2 instances you create.");
    var success = await ec2Methods.AuthorizeSecurityGroupIngress(groupName);

    secGroups = await ec2Methods.DescribeSecurityGroups(secGroupId);
    Console.WriteLine($"Now let's look at the permissions again.");
    secGroups.ForEach(group =>
    {
        ec2Methods.DisplaySecurityGroupInfoAsync(group);
    });
    uiMethods.PressEnter();

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

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

    var images = await ec2Methods.DescribeImages(imageIds);

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

    int choice;
    bool validNumber = false;

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

    var selectedImage = images[choice - 1];

    // Display available instance types.
    uiMethods.DisplayTitle("Instance Types");
    var instanceTypes = await
ec2Methods.DescribeInstanceTypes(selectedImage.Architecture);
```

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

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

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

// Create an EC2 instance.
uiMethods.DisplayTitle("Creating an EC2 Instance");
var instanceId = await ec2Methods.RunInstances(selectedImage.ImageId,
selectedInstanceType, keyPairName, secGroupId);
Console.Write("Waiting for the instance to start.");
var isRunning = false;
do
{
    isRunning = await ec2Methods.WaitForInstanceState(instanceId,
InstanceStateName.Running);
} while (!isRunning);

uiMethods.PressEnter();

var instance = await ec2Methods.DescribeInstance(instanceId);
uiMethods.DisplayTitle("New Instance Information");
ec2Methods.DisplayInstanceInformation(instance);

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

uiMethods.PressEnter();

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

await ec2Methods.StopInstances(instanceId);
```

```
    var hasStopped = false;
    do
    {
        hasStopped = await ec2Methods.WaitForInstanceState(instanceId,
InstanceStateName.Stopped);
    } while (!hasStopped);

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

    Console.WriteLine("Now let's start it up again.");
    await ec2Methods.StartInstances(instanceId);
    Console.WriteLine("Waiting for instance to start. ");

    isRunning = false;
    do
    {
        isRunning = await ec2Methods.WaitForInstanceState(instanceId,
InstanceStateName.Running);
    } while (!isRunning);

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

    instance = await ec2Methods.DescribeInstance(instanceId);
    uiMethods.DisplayTitle("New Instance Information");
    ec2Methods.DisplayInstanceInformation(instance);

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

    uiMethods.PressEnter();

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

    await ec2Methods.StopInstances(instanceId);
    hasStopped = false;
    do
    {
        hasStopped = await ec2Methods.WaitForInstanceState(instanceId,
InstanceStateName.Stopped);
    } while (!hasStopped);
```

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

uiMethods.DisplayTitle("Allocate Elastic IP address");
Console.WriteLine("You can allocate an Elastic IP address and associate
it with your instance\nto keep a consistent IP address even when your instance
restarts.");
var allocationId = await ec2Methods.AllocateAddress();
Console.WriteLine("Now we will associate the Elastic IP address with our
instance.");
var associationId = await ec2Methods.AssociateAddress(allocationId,
instanceId);

// Start the instance again.
Console.WriteLine("Now let's start the instance again.");
await ec2Methods.StartInstances(instanceId);
Console.WriteLine("Waiting for instance to start. ");

isRunning = false;
do
{
    isRunning = await ec2Methods.WaitForInstanceState(instanceId,
InstanceStateName.Running);
} while (!isRunning);

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

instance = await ec2Methods.DescribeInstance(instanceId);
uiMethods.DisplayTitle("Instance information");
ec2Methods.DisplayInstanceInformation(instance);

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

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

await ec2Methods.StopInstances(instanceId);

hasStopped = false;
do
{
```

```
        hasStopped = await ec2Methods.WaitForInstanceState(instanceId,
InstanceStateName.Stopped);
    } while (!hasStopped);

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

    Console.WriteLine("Now let's start it up again.");
    await ec2Methods.StartInstances(instanceId);
    Console.WriteLine("Waiting for instance to start. ");

    isRunning = false;
    do
    {
        isRunning = await ec2Methods.WaitForInstanceState(instanceId,
InstanceStateName.Running);
    } while (!isRunning);

    instance = await ec2Methods.DescribeInstance(instanceId);
    uiMethods.DisplayTitle("New Instance Information");
    ec2Methods.DisplayInstanceInformation(instance);
    Console.WriteLine("Note that the IP address did not change this time.");
    uiMethods.PressEnter();

    uiMethods.DisplayTitle("Clean up resources");

    Console.WriteLine("Now let's clean up the resources we created.");

    // Terminate the instance.
    Console.WriteLine("Terminating the instance we created.");
    var stateChange = await ec2Methods.TerminateInstances(instanceId);

    // Wait for the instance state to be terminated.
    var hasTerminated = false;
    do
    {
        hasTerminated = await ec2Methods.WaitForInstanceState(instanceId,
InstanceStateName.Terminated);
    } while (!hasTerminated);

    Console.WriteLine($"The instance {instanceId} has been terminated.");
    Console.WriteLine("Now we can disassociate the Elastic IP address and
release it.");

    // Disassociate the Elastic IP address.
```

```

    var disassociated = ec2Methods.DisassociateIp(associationId);

    // Delete the Elastic IP address.
    var released = ec2Methods.ReleaseAddress(allocationId);

    // Delete the security group.
    Console.WriteLine($"Deleting the Security Group: {groupName}.");
    success = await ec2Methods.DeleteSecurityGroup(secGroupId);
    if (success)
    {
        Console.WriteLine($"Successfully deleted {groupName}.");
    }

    // Delete the RSA key pair.
    Console.WriteLine($"Deleting the key pair: {keyPairName}");
    await ec2Methods.DeleteKeyPair(keyPairName);
    Console.WriteLine("Deleting the temporary file with the key
information.");
    ec2Methods.DeleteTempFile(tempFileName);
    uiMethods.PressEnter();

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

```

Définissez une classe qui encapsule les actions EC2.

```

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

    public EC2Wrapper(IAmazonEC2 amazonService)
    {
        _amazonEC2 = amazonService;
    }

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

```

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

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

/// <summary>
/// Associate an Elastic IP address to an EC2 instance.
/// </summary>
/// <param name="allocationId">The allocation Id of an Elastic IP address.</
param>
/// <param name="instanceId">The instance Id of the EC2 instance to
/// associate the address with.</param>
/// <returns>The association Id that represents
/// the association of the Elastic IP address with an instance.</returns>
public async Task<string> AssociateAddress(string allocationId, string
instanceId)
{
    var request = new AssociateAddressRequest
    {
        AllocationId = allocationId,
        InstanceId = instanceId
    };

    var response = await _amazonEC2.AssociateAddressAsync(request);
    return response.AssociationId;
}

/// <summary>
/// Authorize the local computer ingress to EC2 instances associated
/// with the virtual private cloud (VPC) security group.
/// </summary>
/// <param name="groupName">The name of the security group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> AuthorizeSecurityGroupIngress(string groupName)
{
    // Get the IP address for the local computer.
    var ipAddress = await GetIpAddress();
    Console.WriteLine($"Your IP address is: {ipAddress}");
}
```

```
    var ipRanges = new List<IpRange> { new IpRange { CidrIp =
    $"{ipAddress}/32" } };
    var permission = new IpPermission
    {
        Ipv4Ranges = ipRanges,
        IpProtocol = "tcp",
        FromPort = 22,
        ToPort = 22
    };
    var permissions = new List<IpPermission> { permission };
    var response = await _amazonEC2.AuthorizeSecurityGroupIngressAsync(
        new AuthorizeSecurityGroupIngressRequest(groupName, permissions));
    return response.HttpStatusCode == HttpStatusCode.OK;
}

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

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

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

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

if (response.HttpStatusCode == HttpStatusCode.OK)
{
    var kp = response.KeyPair;
    return kp;
}
else
{
    Console.WriteLine("Could not create key pair.");
    return null;
}
}

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

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

    return pemFileName;
}

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

```
        new CreateSecurityGroupRequest(groupName, groupDescription));

    return response.GroupId;
}

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

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

/// <summary>
/// Delete an Amazon EC2 key pair.
/// </summary>
/// <param name="keyPairName">The name of the key pair to delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteKeyPair(string keyPairName)
{
    try
    {
        await _amazonEC2.DeleteKeyPairAsync(new
DeleteKeyPairRequest(keyPairName)).ConfigureAwait(false);
    }
}
```

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

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

/// <summary>
/// Delete an Amazon EC2 security group.
/// </summary>
/// <param name="groupName">The name of the group to delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteSecurityGroup(string groupId)
{
    var response = await _amazonEC2.DeleteSecurityGroupAsync(new
DeleteSecurityGroupRequest { GroupId = groupId });
    return response.HttpStatusCode == HttpStatusCode.OK;
}

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

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

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

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

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

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

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

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

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

    /// <summary>
    /// Get information about existing EC2 images.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task DescribeInstances()
    {
        // List all EC2 instances.
        await GetInstanceDescriptions();

        string tagName = "IncludeInList";
        string tagValue = "Yes";
        await GetInstanceDescriptionsFiltered(tagName, tagValue);
    }

    /// <summary>
    /// Get information for all existing Amazon EC2 instances.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task GetInstanceDescriptions()
    {
        Console.WriteLine("Showing all instances:");
        var paginator = _amazonEC2.Paginators.DescribeInstances(new
DescribeInstancesRequest());
```

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

/// <summary>
/// Get information about EC2 instances filtered by a tag name and value.
/// </summary>
/// <param name="tagName">The name of the tag to filter on.</param>
/// <param name="tagValue">The value of the tag to look for.</param>
/// <returns>Async task.</returns>
public async Task GetInstanceDescriptionsFiltered(string tagName, string
tagValue)
{
    // This tag filters the results of the instance list.
    var filters = new List<Filter>
    {
        new Filter
        {
            Name = $"tag:{tagName}",
            Values = new List<string>
            {
                tagValue,
            },
        },
    };
    var request = new DescribeInstancesRequest
    {
        Filters = filters,
    };

    Console.WriteLine("\\nShowing instances with tag: \\\"IncludeInList\\\" set to
\\\"Yes\\\".");
    var paginator = _amazonEC2.Paginators.DescribeInstances(request);
```

```

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

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

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

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

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

    /// <summary>
    /// Display the instance type information returned by
DescribeInstanceTypesAsync.
    /// </summary>
    /// <param name="instanceTypes">The list of instance type information.</
param>

```

```
public void DisplayInstanceTypeInfo(List<InstanceTypeInfo> instanceTypes)
{
    instanceTypes.ForEach(type =>
    {
        Console.WriteLine($"{type.InstanceType}\t{type.MemoryInfo}");
    });
}

/// <summary>
/// Get information about an Amazon EC2 key pair.
/// </summary>
/// <param name="keyPairName">The name of the key pair.</param>
/// <returns>A list of key pair information.</returns>
public async Task<List<KeyPairInfo>> DescribeKeyPairs(string keyPairName)
{
    var request = new DescribeKeyPairsRequest();
    if (!string.IsNullOrEmpty(keyPairName))
    {
        request = new DescribeKeyPairsRequest
        {
            KeyNames = new List<string> { keyPairName }
        };
    }
    var response = await _amazonEC2.DescribeKeyPairsAsync(request);
    return response.KeyPairs.ToList();
}

/// <summary>
/// Retrieve information for an Amazon EC2 security group.
/// </summary>
/// <param name="groupId">The Id of the Amazon EC2 security group.</param>
/// <returns>A list of security group information.</returns>
public async Task<List<SecurityGroup>> DescribeSecurityGroups(string groupId)
{
    var request = new DescribeSecurityGroupsRequest();
    var groupIds = new List<string> { groupId };
    request.GroupIds = groupIds;

    var response = await _amazonEC2.DescribeSecurityGroupsAsync(request);
    return response.SecurityGroups;
}

/// <summary>
```

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

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

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

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

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

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

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

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

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

```
    });
}

/// <summary>
/// Disassociate an Elastic IP address from an EC2 instance.
/// </summary>
/// <param name="associationId">The association Id.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DisassociateIp(string associationId)
{
    var response = await _amazonEC2.DisassociateAddressAsync(
        new DisassociateAddressRequest { AssociationId = associationId });
    return response.HttpStatusCode == HttpStatusCode.OK;
}

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

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

    var response = await _amazonEC2.RebootInstancesAsync(request);
}
```

```
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine("Instances successfully rebooted.");
        }
        else
        {
            Console.WriteLine("Could not reboot one or more instances.");
        }
    }

    /// <summary>
    /// Release an Elastic IP address.
    /// </summary>
    /// <param name="allocationId">The allocation Id of the Elastic IP address.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> ReleaseAddress(string allocationId)
    {
        var request = new ReleaseAddressRequest
        {
            AllocationId = allocationId
        };

        var response = await _amazonEC2.ReleaseAddressAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

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

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

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

    var response = await _amazonEC2.StartInstancesAsync(request);

    if (response.StartingInstances.Count > 0)
    {
        var instances = response.StartingInstances;
        instances.ForEach(i =>
        {
            Console.WriteLine($"Successfully started the EC2 instance with
instance ID: {i.InstanceId}.");
        });
    }
}

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

```
/// <returns>Async task.</returns>
public async Task StopInstances(string ec2InstanceId)
{
    // In addition to the list of instance Ids, the
    // request can also include the following properties:
    //     Force      When true, forces the instances to
    //                 stop but you must check the integrity
    //                 of the file system. Not recommended on
    //                 Windows instances.
    //     Hibernate  When true, hibernates the instance if the
    //                 instance was enabled for hibernation when
    //                 it was launched.
    var request = new StopInstancesRequest
    {
        InstanceIds = new List<string> { ec2InstanceId },
    };

    var response = await _amazonEC2.StopInstancesAsync(request);

    if (response.StoppingInstances.Count > 0)
    {
        var instances = response.StoppingInstances;
        instances.ForEach(i =>
        {
            Console.WriteLine($"Successfully stopped the EC2 Instance " +
                              $"with InstanceID: {i.InstanceId}.");
        });
    }
}

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

```
        var response = await _amazonEC2.TerminateInstancesAsync(request);
        return response.TerminatingInstances;
    }

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

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

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

        return hasState;
    }
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for .NET .
 - [AllocateAddress](#)
 - [AssociateAddress](#)

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

Bash

AWS CLI avec le script Bash

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Exécutez un scénario interactif à une invite de commande.

```
#####  
# function get_started_with_ec2_instances  
#  
# Runs an interactive scenario that shows how to get started using EC2 instances.  
#
```

```

# "EC2 access" permissions are needed to run this code.
#
# Returns:
# 0 - If successful.
# 1 - If an error occurred.
#####
function get_started_with_ec2_instances() {
    # Requires version 4 for mapfile.
    local required_version=4.0

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

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

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

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

            source ./ec2_operations.sh
        fi
    }

    echo_repeat "*" 88

```

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

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

echo -n "Enter a unique name for your key: "
get_input
local key_name
key_name=$get_input_result

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

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

chmod 400 "${key_file_name}"

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

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

```

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

echo "Security group created with ID $security_group_id"
echo

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

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

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

echo "Security group rules updated"

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

mapfile -t parameters <<<"$security_group_description"
IFS=$'\t' read -ra entries <<<"${parameters[0]}"

```

```

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

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

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

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

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

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

if ((images_count == 0)); then
    errecho "No images found. This demo will exit."

```

```

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

echo_repeat "*" 88
echo_repeat "*" 88

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

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

echo "Great choice."
echo

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

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

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

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

integer_input "Which one do you want to use? " 1 "${#instance_types[@]}"

```

```
"
choice=${get_input_result}
local instance_type=${instance_types[$((choice - 1))]}
echo "Another great choice."
echo

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

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

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

echo
print_instance_details "${instance_details}"

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

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

echo_repeat "*" 88
echo_repeat "*" 88

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

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

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

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

print_instance_details "${instance_details}"

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

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

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

echo_repeat "*" 88
echo_repeat "*" 88

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

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

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

echo "Allocated static Elastic IP address: $elastic_ip"

local association_id
association_id=$(ec2_associate_address -i "$instance_id" -a "$allocation_id")
|| {
    errecho "Failed to associate an address. This demo will exit."
```

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

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

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

echo_repeat "*" 88
echo_repeat "*" 88

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

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

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

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

print_instance_details "${instance_details}"

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

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

echo_repeat "*" 88
echo_repeat "*" 88

if yes_no_input "Do you want to delete the resources created in this demo: (y/
n) "; then
    clean_up "$key_name" "$key_file_name" "$security_group_id" "$instance_id" \
```

```

    "$allocation_id" "$association_id"
else
    echo "The following resources were not deleted."
    echo "Key pair: $key_name"
    echo "Key file: $key_file_name"
    echo "Security group: $security_group_id"
    echo "Instance: $instance_id"
    echo "Elastic IP address: $elastic_ip"
fi
}

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

    if [ -n "$association_id" ]; then
        # bashsupport disable=BP2002
        if (ec2_disassociate_address -a "$association_id"); then
            echo "Disassociated elastic IP address with ID $association_id"
        else
            errecho "The elastic IP address disassociation failed."
            result=1
        fi
    fi
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
        esac
    done
}
```

```

        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++))
        fi
    fi
}

```

```

    echo -n "$get_input_result"
else
    echo "MOCK_INPUT_ARRAY has no more elements" 1>&2
    return 1
fi
fi

return 0
}

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

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

```

```
echo

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

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

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

        if ! get_input; then
            return 1
        fi

        input="$get_input_result"

        # Check if the input is a valid integer
        if [[ "$input" =~ ^-[0-9]+$ ]]; then
            # Check if the input is within the specified range
```

```
    if ((input >= min_value && input <= max_value)); then
        return 0
    else
        echo "Error: Input, $input, must be between $min_value and $max_value."
    fi
else
    echo "Error: Invalid input- $input. Please enter an integer."
fi
done
}
#####
# function new_line_and_tab_to_list
#
# This function takes a string input containing newlines and tabs, and
# converts it into a list (array) of elements.
#
# Parameters:
#     $1 - The input string containing newlines and tabs.
#
# Returns:
#     The resulting list (array) is stored in the global variable
#     'list_result'.
#####
function new_line_and_tab_to_list() {
    local input=$1
    export list_result

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

#####
# function echo_repeat
#
# This function prints a string 'n' times to stdout.
#
# Parameters:
#     $1 - The string.
#     $2 - Number of times to print the string.
```

```

#
# Outputs:
#   String 'n' times to stdout.
#
# Returns:
#   0
#####
function echo_repeat() {
    local end=$2
    for ((i = 0; i < end; i++)); do
        echo -n "$1"
    done
    echo
}

```

Fonctions DynamoDB utilisées dans ce scénario.

```

#####
# 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
}
```

```

    esac
done
export OPTIND=1

local response

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

echo "$response"

return 0
}

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

  # Function to display usage information
  function usage() {
    echo "function ec2_create_security_group"
    echo "Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group."
  }
}

```

```
    echo " -n security_group_name - The name of the security group."
    echo " -d security_group_description - The description of the security
group."
    echo ""
}

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

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

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

# Create the security group
response=$(aws ec2 create-security-group \
    --group-name "$security_group_name" \
    --description "$security_group_description" \
    --query "GroupId" \
    --output text) || {
    aws_cli_error_log ${?}
```

```

    errecho "ERROR: AWS reports create-security-group operation failed."
    errecho "$response"
    return 1
}

echo "$response"
return 0
}

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

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

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

```

```

        ;;
    \?)
        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
}

```

```

    esac
done
export OPTIND=1

local aws_cli_args=()

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

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

echo "$response"

return 0
}

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

```

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

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

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

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

```

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

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

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

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

local error_code=$?

rm "$tmp_json_file"

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

```

```

fi

echo "$response"
return 0
}

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

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

    # Retrieve the calling parameters.
    while getopt "i:t:k:s:c:h" option; do

```

```
case "${option}" in
  i) image_id="${OPTARG}" ;;
  t) instance_type="${OPTARG}" ;;
  k) key_pair_name="${OPTARG}" ;;
  s) security_group_id="${OPTARG}" ;;
  c) count="${OPTARG}" ;;
  h)
    usage
    return 0
    ;;
  \?)
    echo "Invalid parameter"
    usage
    return 1
    ;;
esac
done
export OPTIND=1

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

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

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

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

```

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

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

echo "$response"

return 0
}

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

    # bashsupport disable=BP5008
    function usage() {
        echo "function ec2_describe_instances"
    }

```

```

    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
instances."
    echo "  -i instance_id - The ID of the instance to describe (optional)."
```

```

    echo "  -q query - The query to filter the response (optional)."
```

```

    echo "  -h - Display help."
    echo ""
}

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

local aws_cli_args=()

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

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

# shellcheck disable=SC2086
response=$(aws ec2 describe-instances \
    "${aws_cli_args[@]}" \

```

```

$query_arg \
--output text) || {
aws_cli_error_log ${?}
errecho "ERROR: AWS reports describe-instances operation failed.$response"
return 1
}

echo "$response"

return 0
}

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

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

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

```

```

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

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

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

return 0
}

#####
# function ec2_start_instances
#
# This function starts one or more Amazon Elastic Compute Cloud (Amazon EC2)
# instances.
#
# Parameters:
#     -i instance_id - The ID(s) of the instance(s) to start (comma-separated).
#     -h - Display help.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####

```

```
function ec2_start_instances() {
    local instance_ids
    local option OPTARG # Required to use getopt command in a function.

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

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

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

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

```

}

return 0
}

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

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

    # Parse the command-line arguments
    while getopts "d:h" option; do
        case "${option}" in
            d) domain="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
        esac
    done
}

```

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

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

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

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

echo "$response"
return 0
}

#####
# function ec2_associate_address
#
# This function associates an Elastic IP address with an Amazon Elastic Compute
Cloud (Amazon EC2) instance.
#
# Parameters:
```

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

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

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

    # Validate the input parameters
```

```

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

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

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

echo "$response"
return 0
}

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

```

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

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

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

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

return 0
}
```

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

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

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

```

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

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

return 0
}

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

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

```

```

}

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

```

```

        usage
        return 1
    ;;
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
}

```

Fonctions utilitaires utilisées dans ce scénario.

```

#####
# 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
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la Référence des commandes AWS CLI .
 - [AllocateAddress](#)
 - [AssociateAddress](#)
 - [AuthorizeSecurityGroupIngress](#)
 - [CreateKeyPair](#)
 - [CreateSecurityGroup](#)
 - [DeleteKeyPair](#)
 - [DeleteSecurityGroup](#)
 - [DescribeImages](#)
 - [DescribeInstanceTypes](#)
 - [DescribeInstances](#)

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

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Creates an RSA key pair and saves the private key data as a .pem file.
 * 2. Lists key pairs.
 * 3. Creates a security group for the default VPC.
 * 4. Displays security group information.
 * 5. Gets a list of Amazon Linux 2 AMIs and selects one.
 * 6. Gets more information about the image.
 * 7. Gets a list of instance types that are compatible with the selected AMI's
```

```

* architecture.
* 8. Creates an instance with the key pair, security group, AMI, and an
* instance type.
* 9. Displays information about the instance.
* 10. Stops the instance and waits for it to stop.
* 11. Starts the instance and waits for it to start.
* 12. Allocates an Elastic IP address and associates it with the instance.
* 13. Displays SSH connection info for the instance.
* 14. Disassociates and deletes the Elastic IP address.
* 15. Terminates the instance and waits for it to terminate.
* 16. Deletes the security group.
* 17. Deletes the key pair.
*/
public class EC2Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
"-");

    public static void main(String[] args) throws InterruptedException {

        final String usage = ""

            Usage:
                <keyName> <fileName> <groupName> <groupDesc> <vpcId>

            Where:
                keyName - A key pair name (for example, TestKeyPair).\s
                fileName - A file name where the key information is written
to.\s

                groupName - The name of the security group.\s
                groupDesc - The description of the security group.\s
                vpcId - A VPC Id value. You can get this value from the AWS
Management Console.\s
                myIpAddress - The IP address of your development machine.\s

            """;

        if (args.length != 6) {
            System.out.println(usage);
            System.exit(1);
        }

        String keyName = args[0];
        String fileName = args[1];
        String groupName = args[2];

```

```
String groupDesc = args[3];
String vpcId = args[4];
String myIpAddress = args[5];

Region region = Region.US_WEST_2;
Ec2Client ec2 = Ec2Client.builder()
    .region(region)
    .build();

SsmClient ssmClient = SsmClient.builder()
    .region(region)
    .build();

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

System.out.println(DASHES);
System.out.println("1. Create an RSA key pair and save the private key
material as a .pem file.");
createKeyPair(ec2, keyName, fileName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. List key pairs.");
describeKeys(ec2);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Create a security group.");
String groupId = createSecurityGroup(ec2, groupName, groupDesc, vpcId,
myIpAddress);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Display security group info for the newly created
security group.");
describeSecurityGroups(ec2, groupId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Get a list of Amazon Linux 2 AMIs and selects one
with amzn2 in the name.");
String instanceId = getParaValues(ssmClient);
```

```
System.out.println("The instance Id is " + instanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Get more information about an amzn2 image.");
String amiValue = describeImage(ec2, instanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Get a list of instance types.");
String instanceType = getInstanceTypes(ec2);
System.out.println("The instance type is " + instanceType);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Create an instance.");
String newInstanceId = runInstance(ec2, instanceType, keyName, groupName,
amiValue);
System.out.println("The instance Id is " + newInstanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Display information about the running instance.
");
String ipAddress = describeEC2Instances(ec2, newInstanceId);
System.out.println("You can SSH to the instance using this command:");
System.out.println("ssh -i " + fileName + "ec2-user@" + ipAddress);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Stop the instance and use a waiter.");
stopInstance(ec2, newInstanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Start the instance and use a waiter.");
startInstance(ec2, newInstanceId);
ipAddress = describeEC2Instances(ec2, newInstanceId);
System.out.println("You can SSH to the instance using this command:");
System.out.println("ssh -i " + fileName + "ec2-user@" + ipAddress);
System.out.println(DASHES);

System.out.println(DASHES);
```

```
        System.out.println("12. Allocate an Elastic IP address and associate it
with the instance.");
        String allocationId = allocateAddress(ec2);
        System.out.println("The allocation Id value is " + allocationId);
        String associationId = associateAddress(ec2, newInstanceId,
allocationId);
        System.out.println("The associate Id value is " + associationId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Describe the instance again.");
        ipAddress = describeEC2Instances(ec2, newInstanceId);
        System.out.println("You can SSH to the instance using this command:");
        System.out.println("ssh -i " + fileName + "ec2-user@" + ipAddress);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Disassociate and release the Elastic IP
address.");
        disassociateAddress(ec2, associationId);
        releaseEC2Address(ec2, allocationId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Terminate the instance and use a waiter.");
        terminateEC2(ec2, newInstanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Delete the security group.");
        deleteEC2SecGroup(ec2, groupId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Delete the key.");
        deleteKeys(ec2, keyName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("You successfully completed the Amazon EC2
scenario.");
        System.out.println(DASHES);
        ec2.close();
    }
```

```
public static void deleteEC2SecGroup(Ec2Client ec2, String groupId) {
    try {
        DeleteSecurityGroupRequest request =
DeleteSecurityGroupRequest.builder()
        .groupId(groupId)
        .build();

        ec2.deleteSecurityGroup(request);
        System.out.println("Successfully deleted security group with Id " +
groupId);

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

public static void terminateEC2(Ec2Client ec2, String instanceId) {
    try {
        Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();

        TerminateInstancesRequest ti = TerminateInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

        System.out.println("Use an Ec2Waiter to wait for the instance to
terminate. This will take a few minutes.");
        ec2.terminateInstances(ti);
        DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

        WaiterResponse<DescribeInstancesResponse> waiterResponse = ec2Waiter
        .waitUntilInstanceTerminated(instanceRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully started instance " + instanceId);
        System.out.println(instanceId + " is terminated!");

    } catch (Ec2Exception e) {
```

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

public static void deleteKeys(Ec2Client ec2, String keyPair) {
    try {
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
            .keyName(keyPair)
            .build();

        ec2.deleteKeyPair(request);
        System.out.println("Successfully deleted key pair named " + keyPair);

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

public static void releaseEC2Address(Ec2Client ec2, String allocId) {
    try {
        ReleaseAddressRequest request = ReleaseAddressRequest.builder()
            .allocationId(allocId)
            .build();

        ec2.releaseAddress(request);
        System.out.println("Successfully released Elastic IP address " +
allocId);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void disassociateAddress(Ec2Client ec2, String associationId) {
    try {
        DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
            .associationId(associationId)
            .build();

        ec2.disassociateAddress(addressRequest);
        System.out.println("You successfully disassociated the address!");
    }
}
```

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

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

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

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

public static String allocateAddress(Ec2Client ec2) {
    try {
        AllocateAddressRequest allocateRequest =
AllocateAddressRequest.builder()
            .domain(DomainType.VPC)
            .build();

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

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

```
public static void startInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();

    StartInstancesRequest request = StartInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to run.
This will take a few minutes.");
    ec2.startInstances(request);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
    .instanceIds(instanceId)
    .build();

    WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceRunning(instanceRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Successfully started instance " + instanceId);
}

public static void stopInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();

    StopInstancesRequest request = StopInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to stop.
This will take a few minutes.");
    ec2.stopInstances(request);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
    .instanceIds(instanceId)
    .build();

    WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceStopped(instanceRequest);
```

```
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully stopped instance " + instanceId);
    }

    public static String describeEC2Instances(Ec2Client ec2, String
newInstanceId) {
        try {
            String pubAddress = "";
            boolean isRunning = false;
            DescribeInstancesRequest request = DescribeInstancesRequest.builder()
                .instanceIds(newInstanceId)
                .build();

            while (!isRunning) {
                DescribeInstancesResponse response =
ec2.describeInstances(request);
                String state =
response.reservations().get(0).instances().get(0).state().name().name();
                if (state.compareTo("RUNNING") == 0) {
                    System.out.println("Image id is " +
response.reservations().get(0).instances().get(0).imageId());
                    System.out.println(
                        "Instance type is " +
response.reservations().get(0).instances().get(0).instanceType());
                    System.out.println(
                        "Instance state is " +
response.reservations().get(0).instances().get(0).state().name());
                    pubAddress =
response.reservations().get(0).instances().get(0).publicIpAddress();
                    System.out.println("Instance address is " + pubAddress);
                    isRunning = true;
                }
            }
            return pubAddress;
        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static String runInstance(Ec2Client ec2, String instanceType, String
keyName, String groupName,
        String amiId) {
```

```
    try {
        RunInstancesRequest runRequest = RunInstancesRequest.builder()
            .instanceType(instanceType)
            .keyName(keyName)
            .securityGroups(groupName)
            .maxCount(1)
            .minCount(1)
            .imageId(amiId)
            .build();

        System.out.println("Going to start an EC2 instance using a waiter");
        RunInstancesResponse response = ec2.runInstances(runRequest);
        String instanceIdVal = response.instances().get(0).instanceId();
        ec2.waiter().waitUntilInstanceRunning(r ->
r.instanceIds(instanceIdVal));
        System.out.println("Successfully started EC2 instance " +
instanceIdVal + " based on AMI " + amiId);
        return instanceIdVal;

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Get a list of instance types.
public static String getInstanceTypes(Ec2Client ec2) {
    String instanceType;
    try {
        DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
            .maxResults(10)
            .build();

        DescribeInstanceTypesResponse response =
ec2.describeInstanceTypes(typesRequest);
        List<InstanceTypeInfo> instanceTypes = response.instanceTypes();
        for (InstanceTypeInfo type : instanceTypes) {
            System.out.println("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
            System.out.println("Network information is " +
type.networkInfo().toString());
        }
    }
}
```

```
        System.out.println("Instance type is " +
type.instanceType().toString());
        instanceType = type.instanceType().toString();
        if (instanceType.compareTo("t2.2xlarge") == 0){
            return instanceType;
        }
    }

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Display the Description field that corresponds to the instance Id value.
public static String describeImage(Ec2Client ec2, String instanceId) {
    try {
        DescribeImagesRequest imagesRequest = DescribeImagesRequest.builder()
            .imageIds(instanceId)
            .build();

        DescribeImagesResponse response = ec2.describeImages(imagesRequest);
        System.out.println("The description of the first image is " +
response.images().get(0).description());
        System.out.println("The name of the first image is " +
response.images().get(0).name());

        // Return the image Id value.
        return response.images().get(0).imageId();

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Get the Id value of an instance with amzn2 in the name.
public static String getParaValues(SsmClient ssmClient) {
    try {
        GetParametersByPathRequest parameterRequest =
GetParametersByPathRequest.builder()
            .path("/aws/service/ami-amazon-linux-latest")
```

```
        .build());

        GetParametersByPathIterable responses =
ssmClient.getParametersByPathPaginator(parameterRequest);
        for
(ssoftware.amazon.awssdk.services.ssm.model.GetParametersByPathResponse
response : responses) {
            System.out.println("Test " + response.nextToken());
            List<Parameter> parameterList = response.parameters();
            for (Parameter para : parameterList) {
                System.out.println("The name of the para is: " +
para.name());
                System.out.println("The type of the para is: " +
para.type());
                if (filterName(para.name())) {
                    return para.value();
                }
            }
        }

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Return true if the name has amzn2 in it. For example:
// /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-arm64-gp2
private static boolean filterName(String name) {
    String[] parts = name.split("/");
    String myValue = parts[4];
    return myValue.contains("amzn2");
}

public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
    try {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupIds(groupId)
            .build();

        // Use a paginator.
```

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

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

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

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

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

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

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

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

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

public static void describeKeys(Ec2Client ec2) {
    try {
        DescribeKeyPairsResponse response = ec2.describeKeyPairs();
        response.keyPairs().forEach(keyPair -> System.out.printf(
            "Found key pair with name %s " +
            "and fingerprint %s",
            keyPair.keyName(),
            keyPair.keyFingerprint()));
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

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

        CreateKeyPairResponse response = ec2.createKeyPair(request);
        String content = response.keyMaterial();
        BufferedWriter writer = new BufferedWriter(new FileWriter(fileName));
```

```
        writer.write(content);
        writer.close();
        System.out.println("Successfully created key pair named " + keyName);

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

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [AllocateAddress](#)
 - [AssociateAddress](#)
 - [AuthorizeSecurityGroupIngress](#)
 - [CreateKeyPair](#)
 - [CreateSecurityGroup](#)
 - [DeleteKeyPair](#)
 - [DeleteSecurityGroup](#)
 - [DescribeImages](#)
 - [DescribeInstanceTypes](#)
 - [DescribeInstances](#)
 - [DescribeKeyPairs](#)
 - [DescribeSecurityGroups](#)
 - [DisassociateAddress](#)
 - [ReleaseAddress](#)
 - [RunInstances](#)
 - [StartInstances](#)
 - [StopInstances](#)
 - [TerminateInstances](#)
 - [UnmonitorInstances](#)

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Exécutez un scénario interactif à une invite de commande.

```
import { mkdtempSync, writeFileSync, rmSync } from "fs";
import { tmpdir } from "os";
import { join } from "path";
import { get } from "http";

import {
  AllocateAddressCommand,
  AssociateAddressCommand,
  AuthorizeSecurityGroupIngressCommand,
  CreateKeyPairCommand,
  CreateSecurityGroupCommand,
  DeleteKeyPairCommand,
  DeleteSecurityGroupCommand,
  DescribeInstancesCommand,
  DescribeKeyPairsCommand,
  DescribeSecurityGroupsCommand,
  DisassociateAddressCommand,
  EC2Client,
  paginateDescribeImages,
  paginateDescribeInstanceTypes,
  ReleaseAddressCommand,
  RunInstancesCommand,
  StartInstancesCommand,
  StopInstancesCommand,
  TerminateInstancesCommand,
  waitUntilInstanceStatusOk,
  waitUntilInstanceStopped,
  waitUntilInstanceTerminated,
} from "@aws-sdk/client-ec2";
import { paginateGetParametersByPath, SSMClient } from "@aws-sdk/client-ssm";
```

```
import { wrapText } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";

const ec2Client = new EC2Client();
const ssmClient = new SSMClient();

const prompter = new Prompter();
const confirmMessage = "Continue?";
const tmpDirectory = mkdtempSync(join(tmpdir(), "ec2-scenario-tmp"));

const createKeyPair = async (keyPairName) => {
  // Create a key pair in Amazon EC2.
  const { KeyMaterial, KeyPairId } = await ec2Client.send(
    // A unique name for the key pair. Up to 255 ASCII characters.
    new CreateKeyPairCommand({ KeyName: keyPairName }),
  );

  // Save the private key in a temporary location.
  writeFileSync(`${tmpDirectory}/${keyPairName}.pem`, KeyMaterial, {
    mode: 0o400,
  });

  return KeyPairId;
};

const describeKeyPair = async (keyPairName) => {
  const command = new DescribeKeyPairsCommand({
    KeyNames: [keyPairName],
  });
  const { KeyPairs } = await ec2Client.send(command);
  return KeyPairs[0];
};

const createSecurityGroup = async (securityGroupName) => {
  const command = new CreateSecurityGroupCommand({
    GroupName: securityGroupName,
    Description: "A security group for the Amazon EC2 example.",
  });
  const { GroupId } = await ec2Client.send(command);
  return GroupId;
};

const allocateIpAddress = async () => {
  const command = new AllocateAddressCommand({});

```

```
const { PublicIp, AllocationId } = await ec2Client.send(command);
return { PublicIp, AllocationId };
};

const getLocalIpAddress = () => {
  return new Promise((res, rej) => {
    get("http://checkip.amazonaws.com", (response) => {
      let data = "";
      response.on("data", (chunk) => (data += chunk));
      response.on("end", () => res(data.trim()));
    }).on("error", (err) => {
      rej(err);
    });
  });
};

const authorizeSecurityGroupIngress = async (securityGroupId) => {
  const ipAddress = await getLocalIpAddress();
  const command = new AuthorizeSecurityGroupIngressCommand({
    GroupId: securityGroupId,
    IpPermissions: [
      {
        IpProtocol: "tcp",
        FromPort: 22,
        ToPort: 22,
        IpRanges: [{ CidrIp: `${ipAddress}/32` }],
      },
    ],
  });
  await ec2Client.send(command);
  return ipAddress;
};

const describeSecurityGroup = async (securityGroupName) => {
  const command = new DescribeSecurityGroupsCommand({
    GroupNames: [securityGroupName],
  });
  const { SecurityGroups } = await ec2Client.send(command);

  return SecurityGroups[0];
};

const getAmznLinux2AMIs = async () => {
```

```
const AMIs = [];
for await (const page of paginateGetParametersByPath(
  {
    client: ssmClient,
  },
  { Path: "/aws/service/ami-amazon-linux-latest" },
)) {
  page.Parameters.forEach((param) => {
    if (param.Name.includes("amzn2")) {
      AMIs.push(param.Value);
    }
  });
}

const imageDetails = [];

for await (const page of paginateDescribeImages(
  { client: ec2Client },
  { ImageIds: AMIs },
)) {
  imageDetails.push(...(page.Images || []));
}

const choices = imageDetails.map((image, index) => ({
  name: `${image.ImageId} - ${image.Description}`,
  value: index,
}));

/**
 * @type {number}
 */
const selectedIndex = await prompter.select({
  message: "Select an image.",
  choices,
});

return imageDetails[selectedIndex];
};

/**
 * @param {import('@aws-sdk/client-ec2').Image} imageDetails
 */
const getCompatibleInstanceTypes = async (imageDetails) => {
  const paginator = paginateDescribeInstanceTypes(
```

```
{ client: ec2Client, pageSize: 25 },
{
  Filters: [
    {
      Name: "processor-info.supported-architecture",
      Values: [imageDetails.Architecture],
    },
    { Name: "instance-type", Values: ["*.micro", "*.small"] },
  ],
},
);

const instanceTypes = [];

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

const choices = instanceTypes.map((type, index) => ({
  name: `${type.InstanceType} - Memory:${type.MemoryInfo.SizeInMiB}`,
  value: index,
})));

/**
 * @type {number}
 */
const selectedIndex = await prompter.select({
  message: "Select an instance type.",
  choices,
});
return instanceTypes[selectedIndex];
};

const runInstance = async ({
  keyPairName,
  securityGroupId,
  imageId,
  instanceType,
}) => {
  const command = new RunInstancesCommand({
    KeyName: keyPairName,
    SecurityGroupIds: [securityGroupId],
```

```
    ImageId: imageId,
    InstanceType: instanceType,
    MinCount: 1,
    MaxCount: 1,
  });

  const { Instances } = await ec2Client.send(command);
  await waitUntilInstanceStatusOk(
    { client: ec2Client },
    { InstanceIds: [Instances[0].InstanceId] },
  );
  return Instances[0].InstanceId;
};

const describeInstance = async (instanceId) => {
  const command = new DescribeInstancesCommand({
    InstanceIds: [instanceId],
  });

  const { Reservations } = await ec2Client.send(command);
  return Reservations[0].Instances[0];
};

const displaySSHConnectionInfo = ({ publicIp, keyPairName }) => {
  return `ssh -i ${tmpDirectory}/${keyPairName}.pem ec2-user@${publicIp}`;
};

const stopInstance = async (instanceId) => {
  const command = new StopInstancesCommand({ InstanceIds: [instanceId] });
  await ec2Client.send(command);
  await waitUntilInstanceStopped(
    { client: ec2Client },
    { InstanceIds: [instanceId] },
  );
};

const startInstance = async (instanceId) => {
  const startCommand = new StartInstancesCommand({ InstanceIds: [instanceId] });
  await ec2Client.send(startCommand);
  await waitUntilInstanceStatusOk(
    { client: ec2Client },
    { InstanceIds: [instanceId] },
  );
  return await describeInstance(instanceId);
};
```

```
};

const associateAddress = async ({ allocationId, instanceId }) => {
  const command = new AssociateAddressCommand({
    AllocationId: allocationId,
    InstanceId: instanceId,
  });

  const { AssociationId } = await ec2Client.send(command);
  return AssociationId;
};

const disassociateAddress = async (associationId) => {
  const command = new DisassociateAddressCommand({
    AssociationId: associationId,
  });
  try {
    await ec2Client.send(command);
  } catch (err) {
    console.warn(
      `Failed to disassociated address with association id: ${associationId}`,
      err,
    );
  }
};

const releaseAddress = async (allocationId) => {
  const command = new ReleaseAddressCommand({
    AllocationId: allocationId,
  });

  try {
    await ec2Client.send(command);
    console.log(`Address with allocation ID ${allocationId} released.\n`);
  } catch (err) {
    console.log(
      `Failed to release address with allocation id: ${allocationId}.`,
      err,
    );
  }
};

const restartInstance = async (instanceId) => {
  console.log("Stopping instance.");
```

```
    await stopInstance(instanceId);
    console.log("Instance stopped.");
    console.log("Starting instance.");
    const { PublicIpAddress } = await startInstance(instanceId);
    return PublicIpAddress;
  };

const terminateInstance = async (instanceId) => {
  const command = new TerminateInstancesCommand({
    InstanceIds: [instanceId],
  });

  try {
    await ec2Client.send(command);
    await waitUntilInstanceTerminated(
      { client: ec2Client },
      { InstanceIds: [instanceId] },
    );
    console.log(`Instance with ID ${instanceId} terminated.\n`);
  } catch (err) {
    console.warn(`Failed to terminate instance ${instanceId}.`, err);
  }
};

const deleteSecurityGroup = async (securityGroupId) => {
  const command = new DeleteSecurityGroupCommand({
    GroupId: securityGroupId,
  });

  try {
    await ec2Client.send(command);
    console.log(`Security group ${securityGroupId} deleted.\n`);
  } catch (err) {
    console.warn(`Failed to delete security group ${securityGroupId}.`, err);
  }
};

const deleteKeyPair = async (keyPairName) => {
  const command = new DeleteKeyPairCommand({
    KeyName: keyPairName,
  });

  try {
    await ec2Client.send(command);
```

```
    console.log(`Key pair ${keyPairName} deleted.\n`);
  } catch (err) {
    console.warn(`Failed to delete key pair ${keyPairName}.`, err);
  }
};

const deleteTemporaryDirectory = () => {
  try {
    rmSync(tmpDirectory, { recursive: true });
    console.log(`Temporary directory ${tmpDirectory} deleted.\n`);
  } catch (err) {
    console.warn(`Failed to delete temporary directory ${tmpDirectory}.`, err);
  }
};

export const main = async () => {
  const keyPairName = "ec2-scenario-key-pair";
  const securityGroupName = "ec2-scenario-security-group";

  let securityGroupId, ipAllocationId, publicIp, instanceId, associationId;

  console.log(wrapText("Welcome to the Amazon EC2 basic usage scenario."));

  try {
    // Prerequisites
    console.log(
      "Before you launch an instance, you'll need a few things:",
      "\n - A Key Pair",
      "\n - A Security Group",
      "\n - An IP Address",
      "\n - An AMI",
      "\n - A compatible instance type",
      "\n\n I'll go ahead and take care of the first three, but I'll need your
      help for the rest.",
    );

    await prompter.confirm({ message: confirmMessage });

    await createKeyPair(keyPairName);
    securityGroupId = await createSecurityGroup(securityGroupName);
    const { PublicIp, AllocationId } = await allocateIpAddress();
    ipAllocationId = AllocationId;
    publicIp = PublicIp;
    const ipAddress = await authorizeSecurityGroupIngress(securityGroupId);
```

```
const { KeyName } = await describeKeyPair(keyPairName);
const { GroupName } = await describeSecurityGroup(securityGroupName);
console.log(`# created the key pair ${KeyName}.\n`);
console.log(
  `# created the security group ${GroupName}`,
  `and allowed SSH access from ${ipAddress} (your IP).\n`,
);
console.log(`# allocated ${publicIp} to be used for your EC2 instance.\n`);

await prompter.confirm({ message: confirmMessage });

// Creating the instance
console.log(wrapText("Create the instance."));
console.log(
  "You get to choose which image you want. Select an amazon-linux-2 image
from the following:",
);
const imageDetails = await getAmznLinux2AMIs();
const instanceTypeDetails = await getCompatibleInstanceTypes(imageDetails);
console.log("Creating your instance. This can take a few seconds.");
instanceId = await runInstance({
  keyPairName,
  securityGroupId,
  imageId: imageDetails.ImageId,
  instanceType: instanceTypeDetails.InstanceType,
});
const instanceDetails = await describeInstance(instanceId);
console.log(`# instance ${instanceId}.\n`);
console.log(instanceDetails);
console.log(
  `
\nYou should now be able to SSH into your instance from another
terminal:`,
  `
\n${displaySSHConnectionInfo({
  publicIp: instanceDetails.PublicIpAddress,
  keyPairName,
})}`,
);

await prompter.confirm({ message: confirmMessage });

// Understanding the IP address.
console.log(wrapText("Understanding the IP address."));
console.log(
```

```
    "When you stop and start an instance, the IP address will change. I'll
restart your",
    "instance for you. Notice how the IP address changes.",
  );
  const ipAddressAfterRestart = await restartInstance(instanceId);
  console.log(
    `
Instance started. The IP address changed from
${instanceDetails.PublicIpAddress} to ${ipAddressAfterRestart}`,
    `
${displaySSHConnectionInfo({
  publicIp: ipAddressAfterRestart,
  keyPairName,
})}`,
  );
  await prompter.confirm({ message: confirmMessage });
  console.log(
    `If you want to the IP address to be static, you can associate an
allocated`,
    `IP address to your instance. I allocated ${publicIp} for you earlier, and
now I'll associate it to your instance.`,
  );
  associationId = await associateAddress({
    allocationId: ipAllocationId,
    instanceId,
  });
  console.log(
    "Done. Now you should be able to SSH using the new IP.\n",
    `${displaySSHConnectionInfo({ publicIp, keyPairName })}`,
  );
  await prompter.confirm({ message: confirmMessage });
  console.log(
    "I'll restart the server again so you can see the IP address remains the
same.",
  );
  const ipAddressAfterAssociated = await restartInstance(instanceId);
  console.log(
    `Done. Here's your SSH info. Notice the IP address hasn't changed.`,
    `
${displaySSHConnectionInfo({
  publicIp: ipAddressAfterAssociated,
  keyPairName,
})}`,
  );
  await prompter.confirm({ message: confirmMessage });
} catch (err) {
  console.error(err);
}
```

```
} finally {
  // Clean up.
  console.log(wrapText("Clean up."));
  console.log("Now I'll clean up all of the stuff I created.");
  await prompter.confirm({ message: confirmMessage });
  console.log("Cleaning up. Some of these steps can take a bit of time.");
  await disassociateAddress(associationId);
  await terminateInstance(instanceId);
  await releaseAddress(ipAllocationId);
  await deleteSecurityGroup(securityGroupId);
  deleteTemporaryDirectory();
  await deleteKeyPair(keyPairName);
  console.log(
    "Done cleaning up. Thanks for staying until the end!",
    "If you have any feedback please use the feedback button in the docs",
    "or create an issue on GitHub.",
  );
}
};
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for JavaScript .
 - [AllocateAddress](#)
 - [AssociateAddress](#)
 - [AuthorizeSecurityGroupIngress](#)
 - [CreateKeyPair](#)
 - [CreateSecurityGroup](#)
 - [DeleteKeyPair](#)
 - [DeleteSecurityGroup](#)
 - [DescribeImages](#)
 - [DescribeInstanceTypes](#)
 - [DescribeInstances](#)
 - [DescribeKeyPairs](#)
 - [DescribeSecurityGroups](#)
 - [DisassociateAddress](#)
 - [ReleaseAddress](#)

- [RunInstances](#)
- [StartInstances](#)
- [StopInstances](#)
- [TerminateInstances](#)
- [UnmonitorInstances](#)

Kotlin

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**
```

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

```
For more information, see the following documentation topic:
```

```
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
```

```
This Kotlin example performs the following tasks:
```

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

```

13. Displays SSH connection info for the instance.
14. Disassociates and deletes the Elastic IP address.
15. Terminates the instance.
16. Deletes the security group.
17. Deletes the key pair.
*/

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

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

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

        """

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

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

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

        println(DASHES)

```

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

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

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

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

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

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

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

println(DASHES)
println("8. Create an instance.")
if (amiValue != null) {
    newInstanceId = runInstanceSc(instanceType, keyName, groupName, amiValue)
    println("The instance Id is $newInstanceId")
}
println(DASHES)

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

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

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

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

```
println(DASHES)

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

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

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

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

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

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

suspend fun deleteKeysSc(keyPair: String) {
    val request =
        DeleteKeyPairRequest {
            keyName = keyPair
        }
}
```

```
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.deleteKeyPair(request)
        println("Successfully deleted key pair named $keyPair")
    }
}

suspend fun deleteEC2SecGroupSc(groupIdVal: String) {
    val request =
        DeleteSecurityGroupRequest {
            groupId = groupIdVal
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.deleteSecurityGroup(request)
        println("Successfully deleted security group with Id $groupIdVal")
    }
}

suspend fun terminateEC2Sc(instanceIdVal: String) {
    val ti =
        TerminateInstancesRequest {
            instanceIds = listOf(instanceIdVal)
        }
    println("Wait for the instance to terminate. This will take a few minutes.")
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.terminateInstances(ti)
        ec2.waitForInstanceTerminated {
            // suspend call
            instanceIds = listOf(instanceIdVal)
        }
        println("$instanceIdVal is terminated!")
    }
}

suspend fun releaseEC2AddressSc(allocId: String?) {
    val request =
        ReleaseAddressRequest {
            allocationId = allocId
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.releaseAddress(request)
        println("Successfully released Elastic IP address $allocId")
    }
}
```

```
suspend fun disassociateAddressSc(associationIdVal: String?) {
    val addressRequest =
        DisassociateAddressRequest {
            associationId = associationIdVal
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.disassociateAddress(addressRequest)
        println("You successfully disassociated the address!")
    }
}

suspend fun associateAddressSc(
    instanceIdVal: String?,
    allocationIdVal: String?,
): String? {
    val associateRequest =
        AssociateAddressRequest {
            instanceId = instanceIdVal
            allocationId = allocationIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val associateResponse = ec2.associateAddress(associateRequest)
        return associateResponse.associationId
    }
}

suspend fun allocateAddressSc(): String? {
    val allocateRequest =
        AllocateAddressRequest {
            domain = DomainType.Vpc
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val allocateResponse = ec2.allocateAddress(allocateRequest)
        return allocateResponse.allocationId
    }
}

suspend fun startInstanceSc(instanceId: String) {
    val request =
        StartInstancesRequest {
            instanceIds = listOf(instanceId)
        }
}
```

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

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

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

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

    while (!isRunning) {
        Ec2Client { region = "us-west-2" }.use { ec2 ->
            val response = ec2.describeInstances(request)
            val state =
                response.reservations
```

```

        ?.get(0)
        ?.instances
        ?.get(0)
        ?.state
        ?.name
        ?.value
    if (state != null) {
        if (state.compareTo("running") == 0) {
            println("Image id is
${response.reservations!!.get(0).instances?.get(0)?.imageId}")
            println("Instance type is
${response.reservations!!.get(0).instances?.get(0)?.instanceType}")
            println("Instance state is
${response.reservations!!.get(0).instances?.get(0)?.state}")
            pubAddress =
                response.reservations!!
                    .get(0)
                    .instances
                    ?.get(0)
                    ?.publicIpAddress
                    .toString()
            println("Instance address is $pubAddress")
            isRunning = true
        }
    }
}
}
return pubAddress
}

suspend fun runInstanceSc(
    instanceTypeVal: String,
    keyNameVal: String,
    groupNameVal: String,
    amiIdVal: String,
): String {
    val runRequest =
        RunInstancesRequest {
            instanceType = InstanceType.fromValue(instanceTypeVal)
            keyName = keyNameVal
            securityGroups = listOf(groupNameVal)
            maxCount = 1
            minCount = 1
            imageId = amiIdVal

```

```
    }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.runInstances(runRequest)
        val instanceId = response.instances?.get(0)?.instanceId
        println("Successfully started EC2 Instance $instanceId based on AMI
$amiIdVal")
        return instanceId.toString()
    }
}

// Get a list of instance types.
suspend fun getInstanceTypesSc(): String {
    var instanceType = ""
    val filterObs = ArrayList<Filter>()
    val filter =
        Filter {
            name = "processor-info.supported-architecture"
            values = listOf("arm64")
        }

    filterObs.add(filter)
    val typesRequest =
        DescribeInstanceTypesRequest {
            filters = filterObs
            maxResults = 10
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeInstanceTypes(typesRequest)
        response.instanceTypes?.forEach { type ->
            println("The memory information of this type is
${type.memoryInfo?.sizeInMib}")
            println("Maximum number of network cards is
${type.networkInfo?.maximumNetworkCards}")
            instanceType = type.instanceType.toString()
        }
        return instanceType
    }
}

// Display the Description field that corresponds to the instance Id value.
suspend fun describeImageSc(instanceId: String): String? {
    val imagesRequest =
        DescribeImagesRequest {
```

```
        imageIds = listOf(instanceId)
    }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeImages(imagesRequest)
        println("The description of the first image is
        ${response.images?.get(0)?.description}")
        println("The name of the first image is
        ${response.images?.get(0)?.name}")

        // Return the image Id value.
        return response.images?.get(0)?.imageId
    }
}

// Get the Id value of an instance with amzn2 in the name.
suspend fun getParaValuesSc(): String? {
    val parameterRequest =
        GetParametersByPathRequest {
            path = "/aws/service/ami-amazon-linux-latest"
        }

    SsmClient { region = "us-west-2" }.use { ssmClient ->
        val response = ssmClient.getParametersByPath(parameterRequest)
        response.parameters?.forEach { para ->
            println("The name of the para is: ${para.name}")
            println("The type of the para is: ${para.type}")
            println("")
            if (para.name?.let { filterName(it) } == true) {
                return para.value
            }
        }
    }
    return ""
}

fun filterName(name: String): Boolean {
    val parts = name.split("/").toTypedArray()
    val myValue = parts[4]
    return myValue.contains("amzn2")
}

suspend fun describeSecurityGroupsSc(groupId: String) {
    val request =
```

```
DescribeSecurityGroupsRequest {
    groupIds = listOf(groupId)
}

Ec2Client { region = "us-west-2" }.use { ec2 ->
    val response = ec2.describeSecurityGroups(request)
    for (group in response.securityGroups!!) {
        println("Found Security Group with id " + group.groupId.toString() +
" and group VPC " + group.vpcId)
    }
}

suspend fun createEC2SecurityGroupSc(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
    myIpAddress: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val resp = ec2.createSecurityGroup(request)
        val ipRange =
            IpRange {
                cidrIp = "$myIpAddress/0"
            }

        val ipPerm =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 80
                fromPort = 80
                ipRanges = listOf(ipRange)
            }

        val ipPerm2 =
            IpPermission {
                ipProtocol = "tcp"
            }
    }
}
```

```
        toPort = 22
        fromPort = 22
        ipRanges = listOf(ipRange)
    }

    val authRequest =
        AuthorizeSecurityGroupIngressRequest {
            groupName = groupNameVal
            ipPermissions = listOf(ipPerm, ipPerm2)
        }
    ec2.authorizeSecurityGroupIngress(authRequest)
    println("Successfully added ingress policy to Security Group
$groupNameVal")
    return resp.groupId
}
}

suspend fun describeEC2KeysSc() {
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeKeyPairs(DescribeKeyPairsRequest {})
        response.keyPairs?.forEach { keyPair ->
            println("Found key pair with name ${keyPair.keyName} and fingerprint
${ keyPair.keyFingerprint}")
        }
    }
}

suspend fun createKeyPairSc(
    keyNameVal: String,
    fileNameVal: String,
) {
    val request =
        CreateKeyPairRequest {
            keyName = keyNameVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.createKeyPair(request)
        val content = response.keyMaterial
        if (content != null) {
            File(fileNameVal).writeText(content)
        }
        println("Successfully created key pair named $keyNameVal")
    }
}
```

```
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans AWS SDK for Kotlin API reference.
 - [AllocateAddress](#)
 - [AssociateAddress](#)
 - [AuthorizeSecurityGroupIngress](#)
 - [CreateKeyPair](#)
 - [CreateSecurityGroup](#)
 - [DeleteKeyPair](#)
 - [DeleteSecurityGroup](#)
 - [DescribeImages](#)
 - [DescribeInstanceTypes](#)
 - [DescribeInstances](#)
 - [DescribeKeyPairs](#)
 - [DescribeSecurityGroups](#)
 - [DisassociateAddress](#)
 - [ReleaseAddress](#)
 - [RunInstances](#)
 - [StartInstances](#)
 - [StopInstances](#)
 - [TerminateInstances](#)
 - [UnmonitorInstances](#)

Python

SDK pour Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Exécutez un scénario interactif à une invite de commande.

```
class Ec2InstanceScenario:
    """Runs an interactive scenario that shows how to get started using EC2
    instances."""

    def __init__(self, inst_wrapper, key_wrapper, sg_wrapper, eip_wrapper,
                 ssm_client):
        """
        :param inst_wrapper: An object that wraps instance actions.
        :param key_wrapper: An object that wraps key pair actions.
        :param sg_wrapper: An object that wraps security group actions.
        :param eip_wrapper: An object that wraps Elastic IP actions.
        :param ssm_client: A Boto3 AWS Systems Manager client.
        """

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

    @demo_func
    def create_and_list_key_pairs(self):
        """
        1. Creates an RSA key pair and saves its private key data as a .pem file
        in secure temporary storage. The private key data is deleted after the example
        completes.
        2. Lists the first five key pairs for the current account.
        """
        print(
            "Let's create an RSA key pair that you can be use to securely connect
            to "
            "your EC2 instance."
        )
        key_name = q.ask("Enter a unique name for your key: ", q.non_empty)
        self.key_wrapper.create(key_name)
        print(
            f"Created a key pair {self.key_wrapper.key_pair.key_name} and saved
            the "
            f"private key to {self.key_wrapper.key_file_path}.\n"
        )
        if q.ask("Do you want to list some of your key pairs? (y/n) ",
                q.is_yesno):
```

```

        self.key_wrapper.list(5)

    @demo_func
    def create_security_group(self):
        """
        1. Creates a security group for the default VPC.
        2. Adds an inbound rule to allow SSH. The SSH rule allows only
           inbound traffic from the current computer's public IPv4 address.
        3. Displays information about the security group.

        This function uses 'http://checkip.amazonaws.com' to get the current
        public IP
        address of the computer that is running the example. This method works in
        most
        cases. However, depending on how your computer connects to the internet,
        you
        might have to manually add your public IP address to the security group
        by using
        the AWS Management Console.
        """
        print("Let's create a security group to manage access to your instance.")
        sg_name = q.ask("Enter a unique name for your security group: ",
q.non_empty)
        security_group = self.sg_wrapper.create(
            sg_name, "Security group for example: get started with instances."
        )
        print(
            f"Created security group {security_group.group_name} in your default
"
            f"VPC {security_group.vpc_id}.\n"
        )

        ip_response = urllib.request.urlopen("http://checkip.amazonaws.com")
        current_ip_address = ip_response.read().decode("utf-8").strip()
        print("Let's add a rule to allow SSH only from your current IP address.")
        print(f"Your public IP address is {current_ip_address}.")
        q.ask("Press Enter to add this rule to your security group.")
        response = self.sg_wrapper.authorize_ingress(current_ip_address)
        if response["Return"]:
            print("Security group rules updated.")
        else:
            print("Couldn't update security group rules.")
        self.sg_wrapper.describe()

```

```
@demo_func
def create_instance(self):
    """
    1. Gets a list of Amazon Linux 2 AMIs from AWS Systems Manager.
    Specifying the
        '/aws/service/ami-amazon-linux-latest' path returns only the latest
    AMIs.
    2. Gets and displays information about the available AMIs and lets you
    select one.
    3. Gets a list of instance types that are compatible with the selected
    AMI and
        lets you select one.
    4. Creates an instance with the previously created key pair and security
    group,
        and the selected AMI and instance type.
    5. Waits for the instance to be running and then displays its
    information.
    """
    ami_paginator = self.ssm_client.get_paginator("get_parameters_by_path")
    ami_options = []
    for page in ami_paginator.paginate(Path="/aws/service/ami-amazon-linux-
latest"):
        ami_options += page["Parameters"]
    amzn2_images = self.inst_wrapper.get_images(
        [opt["Value"] for opt in ami_options if "amzn2" in opt["Name"]]
    )
    print(
        "Let's create an instance from an Amazon Linux 2 AMI. Here are some
options:"
    )
    image_choice = q.choose(
        "Which one do you want to use? ", [opt.description for opt in
amzn2_images]
    )
    print("Great choice!\n")

    print(
        f"Here are some instance types that support the "
        f"{amzn2_images[image_choice].architecture} architecture of the
image:"
    )
    inst_types = self.inst_wrapper.get_instance_types(
        amzn2_images[image_choice].architecture
    )
```

```
    inst_type_choice = q.choose(
        "Which one do you want to use? ", [it["InstanceType"] for it in
inst_types]
    )
    print("Another great choice.\n")

    print("Creating your instance and waiting for it to start...")
    self.inst_wrapper.create(
        amzn2_images[image_choice],
        inst_types[inst_type_choice]["InstanceType"],
        self.key_wrapper.key_pair,
        [self.sg_wrapper.security_group],
    )
    print(f"Your instance is ready:\n")
    self.inst_wrapper.display()

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

    def _display_ssh_info(self):
        """
        Displays an SSH connection string that can be used to connect to a
running
instance.
        """
        print("To connect, open another command prompt and run the following
command:")
        if self.eip_wrapper.elastic_ip is None:
            print(
                f"\tssh -i {self.key_wrapper.key_file_path} "
                f"ec2-user@{self.inst_wrapper.instance.public_ip_address}"
            )
        else:
            print(
                f"\tssh -i {self.key_wrapper.key_file_path} "
                f"ec2-user@{self.eip_wrapper.elastic_ip.public_ip}"
            )
        q.ask("Press Enter when you're ready to continue the demo.")
```

```
@demo_func
def associate_elastic_ip(self):
    """
    1. Allocates an Elastic IP address and associates it with the instance.
    2. Displays an SSH connection string that uses the Elastic IP address.
    """
    print(
        "You can allocate an Elastic IP address and associate it with your
instance\n"
        "to keep a consistent IP address even when your instance restarts."
    )
    elastic_ip = self.eip_wrapper.allocate()
    print(f"Allocated static Elastic IP address: {elastic_ip.public_ip}.")
    self.eip_wrapper.associate(self.inst_wrapper.instance)
    print(f"Associated your Elastic IP with your instance.")
    print(
        "You can now use SSH to connect to your instance by using the Elastic
IP."
    )
    self._display_ssh_info()

@demo_func
def stop_and_start_instance(self):
    """
    1. Stops the instance and waits for it to stop.
    2. Starts the instance and waits for it to start.
    3. Displays information about the instance.
    4. Displays an SSH connection string. When an Elastic IP address is
associated
with the instance, the IP address stays consistent when the instance
stops
and starts.
    """
    print("Let's stop and start your instance to see what changes.")
    print("Stopping your instance and waiting until it's stopped...")
    self.inst_wrapper.stop()
    print("Your instance is stopped. Restarting...")
    self.inst_wrapper.start()
    print("Your instance is running.")
    self.inst_wrapper.display()
    if self.eip_wrapper.elastic_ip is None:
        print(
```

```
        "Every time your instance is restarted, its public IP address
changes."
    )
    else:
        print(
            "Because you have associated an Elastic IP with your instance,
you can \n"
            "connect by using a consistent IP address after the instance
restarts."
        )
    self._display_ssh_info()

@demo_func
def cleanup(self):
    """
    1. Disassociate and delete the previously created Elastic IP.
    2. Terminate the previously created instance.
    3. Delete the previously created security group.
    4. Delete the previously created key pair.
    """
    print("Let's clean everything up. This example created these resources:")
    print(f"\tElastic IP: {self.eip_wrapper.elastic_ip.allocation_id}")
    print(f"\tInstance: {self.inst_wrapper.instance.id}")
    print(f"\tSecurity group: {self.sg_wrapper.security_group.id}")
    print(f"\tKey pair: {self.key_wrapper.key_pair.name}")
    if q.ask("Ready to delete these resources? (y/n) ", q.is_yesno):
        self.eip_wrapper.disassociate()
        print("Disassociated the Elastic IP from the instance.")
        self.eip_wrapper.release()
        print("Released the Elastic IP.")
        print("Terminating the instance and waiting for it to terminate...")
        self.inst_wrapper.terminate()
        print("Instance terminated.")
        self.sg_wrapper.delete()
        print("Deleted security group.")
        self.key_wrapper.delete()
        print("Deleted key pair.")

def run_scenario(self):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s:
%(message)s")

    print("-" * 88)
    print(
```

```

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

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

    print("\nThanks for watching!")
    print("-" * 88)

if __name__ == "__main__":
    try:
        scenario = Ec2InstanceScenario(
            InstanceWrapper.from_resource(),
            KeyPairWrapper.from_resource(),
            SecurityGroupWrapper.from_resource(),
            ElasticIpWrapper.from_resource(),
            boto3.client("ssm"),
        )
        scenario.run_scenario()
    except Exception:
        logging.exception("Something went wrong with the demo.")

```

Définissez une classe qui encapsule les actions de la paire de clés.

```

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

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

```

```

        :param key_file_dir: The folder where the private key information is
        stored.

            This should be a secure folder.
        :param key_pair: A Boto3 KeyPair object. This is a high-level object that
            wraps key pair actions.
        """
        self.ec2_resource = ec2_resource
        self.key_pair = key_pair
        self.key_file_path = None
        self.key_file_dir = key_file_dir

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

    def create(self, key_name):
        """
        Creates a key pair that can be used to securely connect to an EC2
        instance.

        The returned key pair contains private key information that cannot be
        retrieved
        again. The private key data is stored as a .pem file.

        :param key_name: The name of the key pair to create.
        :return: A Boto3 KeyPair object that represents the newly created key
        pair.
        """
        try:
            self.key_pair = self.ec2_resource.create_key_pair(KeyName=key_name)
            self.key_file_path = os.path.join(
                self.key_file_dir.name, f"{self.key_pair.name}.pem"
            )
            with open(self.key_file_path, "w") as key_file:
                key_file.write(self.key_pair.key_material)
        except ClientError as err:
            logger.error(
                "Couldn't create key %s. Here's why: %s: %s",
                key_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

```

```
        else:
            return self.key_pair

    def list(self, limit):
        """
        Displays a list of key pairs for the current account.

        :param limit: The maximum number of key pairs to list.
        """
        try:
            for kp in self.ec2_resource.key_pairs.limit(limit):
                print(f"Found {kp.key_type} key {kp.name} with fingerprint:")
                print(f"\t{kp.key_fingerprint}")
        except ClientError as err:
            logger.error(
                "Couldn't list key pairs. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    def delete(self):
        """
        Deletes a key pair.
        """
        if self.key_pair is None:
            logger.info("No key pair to delete.")
            return

        key_name = self.key_pair.name
        try:
            self.key_pair.delete()
            self.key_pair = None
        except ClientError as err:
            logger.error(
                "Couldn't delete key %s. Here's why: %s : %s",
                key_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Définissez une classe qui encapsule les actions du groupe de sécurité.

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

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

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

    def create(self, group_name, group_description):
        """
        Creates a security group in the default virtual private cloud (VPC) of
        the
        current account.

        :param group_name: The name of the security group to create.
        :param group_description: The description of the security group to
        create.
        :return: A Boto3 SecurityGroup object that represents the newly created
        security group.
        """
        try:
            self.security_group = self.ec2_resource.create_security_group(
```

```
        GroupName=group_name, Description=group_description
    )
except ClientError as err:
    logger.error(
        "Couldn't create security group %s. Here's why: %s: %s",
        group_name,
        err.response["Error"]["Code"],
        err.response["Error"]["Message"],
    )
    raise
else:
    return self.security_group

def authorize_ingress(self, ssh_ingress_ip):
    """
    Adds a rule to the security group to allow access to SSH.

    :param ssh_ingress_ip: The IP address that is granted inbound access to
connect
                           to port 22 over TCP, used for SSH.
    :return: The response to the authorization request. The 'Return' field of
the
            response indicates whether the request succeeded or failed.
    """
    if self.security_group is None:
        logger.info("No security group to update.")
        return

    try:
        ip_permissions = [
            {
                # SSH ingress open to only the specified IP address.
                "IpProtocol": "tcp",
                "FromPort": 22,
                "ToPort": 22,
                "IpRanges": [{"CidrIp": f"{ssh_ingress_ip}/32"}],
            }
        ]
        response = self.security_group.authorize_ingress(
            IpPermissions=ip_permissions
        )
    except ClientError as err:
        logger.error(
```

```
        "Couldn't authorize inbound rules for %s. Here's why: %s: %s",
        self.security_group.id,
        err.response["Error"]["Code"],
        err.response["Error"]["Message"],
    )
    raise
else:
    return response

def describe(self):
    """
    Displays information about the security group.
    """
    if self.security_group is None:
        logger.info("No security group to describe.")
        return

    try:
        print(f"Security group: {self.security_group.group_name}")
        print(f"\tID: {self.security_group.id}")
        print(f"\tVPC: {self.security_group.vpc_id}")
        if self.security_group.ip_permissions:
            print(f"Inbound permissions:")
            pp(self.security_group.ip_permissions)
    except ClientError as err:
        logger.error(
            "Couldn't get data for security group %s. Here's why: %s: %s",
            self.security_group.id,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise

def delete(self):
    """
    Deletes the security group.
    """
    if self.security_group is None:
        logger.info("No security group to delete.")
        return

    group_id = self.security_group.id
```

```

try:
    self.security_group.delete()
except ClientError as err:
    logger.error(
        "Couldn't delete security group %s. Here's why: %s: %s",
        group_id,
        err.response["Error"]["Code"],
        err.response["Error"]["Message"],
    )
    raise

```

Définissez une classe qui encapsule les actions de l'instance.

```

class InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance
    actions."""

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

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

    def create(self, image, instance_type, key_pair, security_groups=None):
        """
        Creates a new EC2 instance. The instance starts immediately after

```

```

        it is created.

        The instance is created in the default VPC of the current account.

        :param image: A Boto3 Image object that represents an Amazon Machine
Image (AMI)
                that defines attributes of the instance that is created.
The AMI
                defines things like the kind of operating system and the
type of
                storage used by the instance.
        :param instance_type: The type of instance to create, such as 't2.micro'.
                The instance type defines things like the number of
CPUs and
                the amount of memory.
        :param key_pair: A Boto3 KeyPair or KeyPairInfo object that represents
the key
                pair that is used to secure connections to the instance.
        :param security_groups: A list of Boto3 SecurityGroup objects that
represents the
                security groups that are used to grant access to
the
                instance. When no security groups are specified,
the
                default security group of the VPC is used.
        :return: A Boto3 Instance object that represents the newly created
instance.
        """
        try:
            instance_params = {
                "ImageId": image.id,
                "InstanceType": instance_type,
                "KeyName": key_pair.name,
            }
            if security_groups is not None:
                instance_params["SecurityGroupIds"] = [sg.id for sg in
security_groups]
            self.instance = self.ec2_resource.create_instances(
                **instance_params, MinCount=1, MaxCount=1
            )[0]
            self.instance.wait_until_running()
        except ClientError as err:
            logging.error(

```

```
        "Couldn't create instance with image %s, instance type %s, and
key %s. "
        "Here's why: %s: %s",
        image.id,
        instance_type,
        key_pair.name,
        err.response["Error"]["Code"],
        err.response["Error"]["Message"],
    )
    raise
else:
    return self.instance

def display(self, indent=1):
    """
    Displays information about an instance.

    :param indent: The visual indent to apply to the output.
    """
    if self.instance is None:
        logger.info("No instance to display.")
        return

    try:
        self.instance.load()
        ind = "\t" * indent
        print(f"{ind}ID: {self.instance.id}")
        print(f"{ind}Image ID: {self.instance.image_id}")
        print(f"{ind}Instance type: {self.instance.instance_type}")
        print(f"{ind}Key name: {self.instance.key_name}")
        print(f"{ind}VPC ID: {self.instance.vpc_id}")
        print(f"{ind}Public IP: {self.instance.public_ip_address}")
        print(f"{ind}State: {self.instance.state['Name']}")
    except ClientError as err:
        logger.error(
            "Couldn't display your instance. Here's why: %s: %s",
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise

def terminate(self):
```

```
"""
Terminates an instance and waits for it to be in a terminated state.
"""
if self.instance is None:
    logger.info("No instance to terminate.")
    return

instance_id = self.instance.id
try:
    self.instance.terminate()
    self.instance.wait_until_terminated()
    self.instance = None
except ClientError as err:
    logging.error(
        "Couldn't terminate instance %s. Here's why: %s: %s",
        instance_id,
        err.response["Error"]["Code"],
        err.response["Error"]["Message"],
    )
    raise

def start(self):
    """
    Starts an instance and waits for it to be in a running state.

    :return: The response to the start request.
    """
    if self.instance is None:
        logger.info("No instance to start.")
        return

    try:
        response = self.instance.start()
        self.instance.wait_until_running()
    except ClientError as err:
        logging.error(
            "Couldn't start instance %s. Here's why: %s: %s",
            self.instance.id,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
    else:
```

```
        return response

def stop(self):
    """
    Stops an instance and waits for it to be in a stopped state.

    :return: The response to the stop request.
    """
    if self.instance is None:
        logger.info("No instance to stop.")
        return

    try:
        response = self.instance.stop()
        self.instance.wait_until_stopped()
    except ClientError as err:
        logger.error(
            "Couldn't stop instance %s. Here's why: %s: %s",
            self.instance.id,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
    else:
        return response

def get_images(self, image_ids):
    """
    Gets information about Amazon Machine Images (AMIs) from a list of AMI
    IDs.

    :param image_ids: The list of AMIs to look up.
    :return: A list of Boto3 Image objects that represent the requested AMIs.
    """
    try:
        images = list(self.ec2_resource.images.filter(ImageIds=image_ids))
    except ClientError as err:
        logger.error(
            "Couldn't get images. Here's why: %s: %s",
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
    )
```

```
        raise
    else:
        return images

def get_instance_types(self, architecture):
    """
    Gets instance types that support the specified architecture and are
    designated
    as either 'micro' or 'small'. When an instance is created, the instance
    type
    you specify must support the architecture of the AMI you use.

    :param architecture: The kind of architecture the instance types must
    support,
                        such as 'x86_64'.
    :return: A list of instance types that support the specified architecture
    and are either 'micro' or 'small'.
    """
    try:
        inst_types = []
        it_paginator = self.ec2_resource.meta.client.get_paginator(
            "describe_instance_types"
        )
        for page in it_paginator.paginate(
            Filters=[
                {
                    "Name": "processor-info.supported-architecture",
                    "Values": [architecture],
                },
                {"Name": "instance-type", "Values": ["*.micro", "*.small"]},
            ]
        ):
            inst_types += page["InstanceTypes"]
    except ClientError as err:
        logger.error(
            "Couldn't get instance types. Here's why: %s: %s",
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
    else:
        return inst_types
```

Définissez une classe qui encapsule les actions de l'adresse IP Elastic.

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

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

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

    def allocate(self):
        """
        Allocates an Elastic IP address that can be associated with an Amazon EC2
        instance. By using an Elastic IP address, you can keep the public IP
        address
        constant even when you restart the associated instance.

        :return: The newly created Elastic IP object. By default, the address is
        not
                               associated with any instance.
        """
        try:
            response =
self.ec2_resource.meta.client.allocate_address(Domain="vpc")
```

```
        self.elastic_ip =
self.ec2_resource.VpcAddress(response["AllocationId"])
    except ClientError as err:
        logger.error(
            "Couldn't allocate Elastic IP. Here's why: %s: %s",
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
    else:
        return self.elastic_ip

def associate(self, instance):
    """
    Associates an Elastic IP address with an instance. When this association
    is
    created, the Elastic IP's public IP address is immediately used as the
    public
    IP address of the associated instance.

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

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

```
def disassociate(self):
    """
    Removes an association between an Elastic IP address and an instance.
    When the
    association is removed, the instance is assigned a new public IP address.
    """
    if self.elastic_ip is None:
        logger.info("No Elastic IP to disassociate.")
        return

    try:
        self.elastic_ip.association.delete()
    except ClientError as err:
        logger.error(
            "Couldn't disassociate Elastic IP %s from its instance. Here's
            why: %s: %s",
            self.elastic_ip.allocation_id,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise

def release(self):
    """
    Releases an Elastic IP address. After the Elastic IP address is released,
    it can no longer be used.
    """
    if self.elastic_ip is None:
        logger.info("No Elastic IP to release.")
        return

    try:
        self.elastic_ip.release()
    except ClientError as err:
        logger.error(
            "Couldn't release Elastic IP address %s. Here's why: %s: %s",
            self.elastic_ip.allocation_id,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans AWS SDK for Python (Boto3) API Reference.
 - [AllocateAddress](#)
 - [AssociateAddress](#)
 - [AuthorizeSecurityGroupIngress](#)
 - [CreateKeyPair](#)
 - [CreateSecurityGroup](#)
 - [DeleteKeyPair](#)
 - [DeleteSecurityGroup](#)
 - [DescribeImages](#)
 - [DescribeInstanceTypes](#)
 - [DescribeInstances](#)
 - [DescribeKeyPairs](#)
 - [DescribeSecurityGroups](#)
 - [DisassociateAddress](#)
 - [ReleaseAddress](#)
 - [RunInstances](#)
 - [StartInstances](#)
 - [StopInstances](#)
 - [TerminateInstances](#)
 - [UnmonitorInstances](#)

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Création de ressources Amazon EC2 à l'aide d'un SDK AWS](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Surveillez les demandes d'API Amazon EC2 à l'aide d'Amazon CloudWatch

Vous pouvez surveiller les demandes d'API Amazon EC2 à l'aide d'Amazon CloudWatch, qui collecte les données brutes et les transforme en indicateurs lisibles en temps quasi réel. Ces statistiques fournissent un moyen simple de suivre l'utilisation et les résultats des opérations de l'API Amazon EC2 au fil du temps. Ces informations vous donnent une meilleure idée des performances de vos applications Web et vous permettent d'identifier et de diagnostiquer divers problèmes. Vous pouvez également définir des alarmes qui surveillent certains seuils et envoyer des notifications ou prendre des mesures spécifiques lorsque ces seuils sont atteints.

Pour plus d'informations à ce sujet CloudWatch, consultez le [guide de CloudWatch l'utilisateur Amazon](#).

Important

Les métriques de l'API Amazon EC2 sont une fonctionnalité optionnelle. Vous devez demander l'accès à cette fonctionnalité. Pour plus d'informations, consultez [the section called "Activer les métriques de l'API Amazon EC2"](#).

Table des matières

- [Activer les métriques de l'API Amazon EC2](#)
- [Mesures et dimensions de l'API Amazon EC2](#)
- [Conservation des données métriques](#)
- [Suivi des demandes effectuées en votre nom](#)
- [Facturation](#)
- [Travailler avec Amazon CloudWatch](#)

Activer les métriques de l'API Amazon EC2

Utilisez la procédure suivante pour demander l'accès à cette fonctionnalité pour votre Compte AWS.

Pour demander l'accès à cette fonctionnalité

1. [AWS Support Centre](#) ouvert.
2. Choisissez Create case (Créer une demande).
3. Choisissez Compte et facturation.
4. Pour le service, sélectionnez Informations générales et Mise en route.
5. Dans Catégorie, choisissez Utilisation AWS et services.
6. Choisissez Next step: Additional information (Étape suivante : informations supplémentaires).
7. Pour Subject (Objet), saisissez **Request access to Amazon EC2 API metrics**.
8. Pour Description, saisissez **Please grant my account access to Amazon EC2 API metrics. Related page: <https://docs.aws.amazon.com/AWSEC2/latest/APIReference/monitor.html>**. Indiquez également la région à laquelle vous devez avoir accès.
9. Choisissez Next step: Solve now or contact us (Étape suivante : résolvez maintenant ou contactez-nous).
10. Dans l'onglet Contactez-nous, choisissez la langue de contact et le mode de contact que vous préférez.
11. Sélectionnez Envoyer.

Mesures et dimensions de l'API Amazon EC2

Métriques

Les métriques de l'API Amazon EC2 sont contenues dans l'AWS/EC2/API espace de noms. Les tableaux suivants répertorient les métriques disponibles pour les demandes d'API Amazon EC2.

Métrique	Description
ClientErrors	<p>Le nombre de demandes d'API ayant échoué en raison d'erreurs du client.</p> <p>Ces erreurs sont généralement causées par une action du client, telle que la spécification d'un paramètre incorrect ou non valide dans la demande,</p>

Métrique	Description
	<p>ou l'utilisation d'une action ou d'une ressource pour le compte d'un utilisateur qui n'est pas autorisé à utiliser l'action ou la ressource.</p> <p>Unité : nombre</p>
RequestLimitExceeded	<p>Le nombre de fois que le taux de demandes maximal autorisé par les API Amazon EC2 a été dépassé pour votre compte.</p> <p>Les demandes d'API Amazon EC2 sont limitées afin de maintenir les performances du service. Si vos demandes ont été limitées, le message d'erreur s'affiche. <code>Client.RequestLimitExceeded</code></p> <p>Unité : nombre</p>
ServerError	<p>Le nombre de demandes d'API ayant échoué en raison d'erreurs internes au serveur.</p> <p>Ces erreurs sont généralement causées par une erreur, une exception ou une défaillance AWS côté serveur.</p> <p>Unité : nombre</p>
SuccessfulCalls	<p>Le nombre de demandes d'API réussies.</p> <p>Unité : nombre</p>

Dimensions

Les données métriques Amazon EC2 peuvent être filtrées pour toutes les actions de l'API EC2. Pour plus d'informations sur les dimensions, consultez [Amazon CloudWatch Concepts](#).

Conservation des données métriques

Les métriques de l'API Amazon EC2 sont envoyées à des CloudWatch intervalles d'une minute. CloudWatch conserve les données métriques comme suit :

- Les points de données d'une durée de 60 secondes (1 minute) sont disponibles pendant 15 jours
- Les points de données d'une durée de 300 secondes (5 minutes) sont disponibles pendant 63 jours.
- Les points de données d'une période de 3 600 secondes (1 heure) sont disponibles pendant 455 jours (15 mois).

Suivi des demandes effectuées en votre nom

Les demandes d'API effectuées par les AWS services en votre nom, telles que les demandes effectuées par des rôles liés à un service, ne sont pas prises en compte dans le calcul de vos limites d'API et n'envoient pas de statistiques à Amazon CloudWatch pour votre compte. Ces demandes ne peuvent pas être surveillées à l'aide de CloudWatch.

Les demandes d'API effectuées en votre nom par des fournisseurs de services tiers sont prises en compte dans le calcul de vos limites de limitation des API et ils envoient des métriques à Amazon CloudWatch pour votre compte. Ces demandes peuvent être surveillées à l'aide de CloudWatch.

Facturation

CloudWatch Les prix et frais standard s'appliquent. Aucun frais supplémentaire n'est appliqué pour l'utilisation des métriques de l'API Amazon EC2. Pour plus d'informations, consultez [Amazon CloudWatch Pricing](#).

Travailler avec Amazon CloudWatch

Table des matières

- [Afficher CloudWatch les métriques](#)
- [Création d' CloudWatch alarmes](#)

Afficher CloudWatch les métriques

Utilisez la procédure suivante pour consulter les métriques de l'API Amazon EC2.

Prérequis

Vous devez activer l'accès aux métriques de l'API Amazon EC2 pour votre compte. Pour plus d'informations, consultez [the section called “Activer les métriques de l'API Amazon EC2”](#).

Pour consulter les métriques de l'API Amazon EC2 à l'aide de la console

1. Ouvrez la CloudWatch console à l'[adresse https://console.aws.amazon.com/cloudwatch/](https://console.aws.amazon.com/cloudwatch/).
2. Dans le volet de navigation, choisissez Metrics, All metrics.
3. Dans l'onglet Parcourir, choisissez l'espace de noms de métriques EC2/API.
4. Pour afficher les métriques, sélectionnez la dimension de métrique.

Pour consulter les métriques de l'API Amazon EC2 à l'aide de la ligne de commande

Utilisez l'une des commandes suivantes :

- [liste-métriques](#) (AWS CLI)

```
aws cloudwatch list-metrics --namespace "AWS/EC2/API"
```

- [Get-CW \(\) MetricList](#) AWS Tools for Windows PowerShell

```
Get-CWMetricList -Namespace "AWS/EC2/API"
```

Création d' CloudWatch alarmes

Vous pouvez créer une CloudWatch alarme qui envoie un message Amazon SNS lorsque l'alarme change d'état. Une alarme surveille une seule métrique pendant la période que vous spécifiez. Il envoie une notification à un sujet SNS en fonction de la valeur de la métrique par rapport à un seuil donné sur un certain nombre de périodes.

Par exemple, vous pouvez créer une alarme qui surveille le nombre de demandes d' DescribeInstancesAPI qui échouent en raison d'erreurs côté serveur. L'alarme suivante envoie une

notification par e-mail lorsque le nombre d'échecs de demandes d' DescribeInstances API atteint un seuil de 10 erreurs côté serveur sur une période de 5 minutes.

Prérequis

Vous devez activer l'accès aux métriques de l'API Amazon EC2 pour votre compte. Pour plus d'informations, consultez [the section called “Activer les métriques de l'API Amazon EC2”](#).

Pour créer une alarme en cas d'erreur du serveur de requêtes d' DescribeInstances API Amazon EC2

1. Ouvrez la CloudWatch console à l'[adresse https://console.aws.amazon.com/cloudwatch/](https://console.aws.amazon.com/cloudwatch/).
2. Dans le panneau de navigation, choisissez Alarms (alertes), All alarms (Toutes les alertes).
3. Choisissez Create alarm (Créer une alerte).
4. Choisissez Sélectionner une métrique, puis spécifiez les éléments suivants :
 - a. Choisissez EC2/API.
 - b. Choisissez Per-Action Metrics.
 - c. Cochez la case à côté de celle-ci DescribeInstances qui se trouve sur la même ligne que le nom de la ServerErrors métrique.
 - d. Choisissez Select metric (Sélectionner une métrique).
5. La page Specify metric and conditions (Spécifier les métriques et les conditions) apparaît, présentant un graphique et d'autres informations sur la métrique et la statistique que vous avez sélectionnées.
 - a. Sous Métrique, spécifiez les éléments suivants :
 - i. Pour Statistics (Statistique), choisissez Sum (Somme).
 - ii. Pour Période, vérifiez que 5 minutes sont sélectionnées.
 - b. Sous Conditions, spécifiez les éléments suivants :
 - i. Pour Threshold type (Type de seuil), choisissez Static (Statique).
 - ii. Pour Whenever ServerErrors is, choisissez Greater/Equal >=.
 - iii. Pour que... , entrez 10.
 - c. Choisissez Suivant.
6. La page Configure actions (Configurer des actions) apparaît.
 - Sous Notification, spécifiez les éléments suivants :

- i. Pour le déclencheur de l'état d'alarme, choisissez En alarme.
 - ii. Pour Sélectionner un sujet SNS, choisissez Sélectionner un sujet SNS existant ou Créer un nouveau sujet, puis complétez les champs obligatoires pour la notification.
 - iii. Choisissez Suivant.
7. La page Ajouter un nom et une description apparaît.
 - a. Dans Nom de l'alarme, entrez le nom de votre alarme. Le nom ne doit contenir que des caractères ASCII.
 - b. Dans Description de l'alarme, entrez une description facultative pour votre alarme.
 - c. Choisissez Suivant.
8. La page de prévisualisation et de création apparaît. Vérifiez que les informations sont correctes, puis choisissez Créer une alarme.

Pour plus d'informations, consultez la section [Utilisation des CloudWatch alarmes Amazon](#) dans le guide de CloudWatch l'utilisateur Amazon.

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.