Guide de l'utilisateur

# AWS Tools for PowerShell

# AWS Tools for PowerShell: Guide de l'utilisateur

# Table of Contents

# Qu'est-ce que les AWS Tools for PowerShell ?

AWS Tools for PowerShell Il s'agit d'un ensemble de PowerShell modules basés sur les fonctionnalités exposées par le AWS SDK for .NET. Ils vous AWS Tools for PowerShell permettent de scripter des opérations sur vos AWS ressources à partir de la ligne de PowerShell commande.

Les applets de commande fournissent une PowerShell expérience idiomatique pour spécifier les paramètres et gérer les résultats, même s'ils sont implémentés à l'aide des différentes requêtes de service. AWS HTTP APIs Par exemple, les applets de commande pour le PowerShell pipeline de AWS Tools for PowerShell support, c'est-à-dire que vous pouvez diriger des PowerShell objets vers et hors des cmdlets.

Ils AWS Tools for PowerShell sont flexibles dans la manière dont ils vous permettent de gérer les informations d'identification, y compris la prise en charge de l'infrastructure AWS Identity and Access Management (IAM). Vous pouvez utiliser les outils avec des informations IAM d'identification utilisateur, des jetons de sécurité temporaires et IAM des rôles.

Ils AWS Tools for PowerShell prennent en charge le même ensemble de services et de AWS régions que ceux pris en charge par leSDK. Vous pouvez l'installer AWS Tools for PowerShell sur des ordinateurs exécutant les systèmes d'exploitation Windows, Linux ou macOS.

> **Note**
>
> AWS Tools for PowerShell la version 4 est la dernière version majeure et est une mise à jour rétrocompatible de la version 3.3. AWS Tools for PowerShell Elle apporte des améliorations significatives tout en maintenant le comportement existant de l'applet de commande. Vos scripts existants devraient continuer à fonctionner après la mise à niveau vers la nouvelle version. Toutefois, nous vous recommandons de les tester soigneusement avant de procéder à la mise à niveau. Pour plus d'informations sur les modifications de la version 4, consultez Migration de la AWS Tools for PowerShell version 3.3 vers la version 4.

Ils AWS Tools for PowerShell sont disponibles sous la forme des trois packages distincts suivants :

- `AWS.Tools`
- AWSPowerShell.NetCore
- AWSPowerShell

# Maintenance et support pour les versions SDK majeures

Pour plus d'informations sur la maintenance et le support SDK des versions majeures et de leurs dépendances sous-jacentes, consultez les informations suivantes dans le guide de référence AWS SDKs and Tools :

- AWS SDKset politique de maintenance des outils
- AWS SDKset matrice de support des versions d'outils

# **`AWS.Tools`**- Une version modularisée du AWS Tools for PowerShell




Cette version de AWS Tools for PowerShell est la version recommandée pour tout ordinateur fonctionnant PowerShell dans un environnement de production. Étant donné qu'elle est modulaire, vous devez télécharger et charger uniquement les modules pour les services que vous souhaitez utiliser. Ceci réduit les délais de téléchargement ainsi que l'utilisation de la mémoire et, dans la plupart des cas, permet l'importation automatique des applets de commande `AWS.Tools`, avec la nécessité d'appeler manuellement `Import-Module` en premier lieu.

Il s'agit de la dernière version de AWS Tools for PowerShell et fonctionne sur tous les systèmes d'exploitation pris en charge, notamment Windows, Linux et macOS. Ce package fournit un module d'installation`AWS.Tools.Installer`, un module commun et un module pour chaque AWS service, par exemple `AWS.Tools.EC2``AWS.Tools.IdentityManagement`,`AWS.Tools.S3`,,, etc. `AWS.Tools.Common`

Le `AWS.Tools.Installer` module fournit des applets de commande qui vous permettent d'installer, de mettre à jour et de supprimer les modules pour chacun des AWS services. Les applets de commande de ce module garantissent automatiquement que vous disposez de tous les modules dépendants nécessaires pour prendre en charge les modules que vous souhaitez utiliser.

Le module `AWS.Tools.Common` fournit des applets de commande pour la configuration et l'authentification, qui ne sont pas spécifiques au service. Pour utiliser les applets de commande

pour un AWS service, il suffit d'exécuter la commande. PowerShellimporte automatiquement le `AWS.Tools.Common` module et le module du AWS service dont vous souhaitez exécuter l'applet de commande. Ce module est automatiquement installé si vous utilisez le module `AWS.Tools.Installer` pour installer les modules de services.

Vous pouvez installer cette version de AWS Tools for PowerShell sur les ordinateurs qui exécutent :

- PowerShell Core 6.0 ou version ultérieure sous Windows, Linux ou macOS.
- Windows PowerShell 5.1 ou version ultérieure sous Windows avec le. NETFramework 4.7.2 ou version ultérieure.

Tout au long de ce guide, lorsque nous aurons besoin de spécifier cette version seulement, nous lui ferons référence sous son nom de module : *`AWS.Tools`*.

# AWSPowerShell. NetCore - Une version mono-module du AWS Tools for PowerShell



Cette version consiste en un seul module volumineux qui prend en charge tous les AWS services. Pour pouvoir utiliser ce module, vous devez l'importer manuellement.

Vous pouvez installer cette version de AWS Tools for PowerShell sur les ordinateurs qui exécutent :

- PowerShell Core 6.0 ou version ultérieure sous Windows, Linux ou macOS.
- Windows PowerShell 3.0 ou version ultérieure sous Windows avec le. NETFramework 4.7.2 ou version ultérieure.

Tout au long de ce guide, lorsque nous devons spécifier cette version uniquement, nous la désignons par son nom de module : AWSPowerShell. NetCore.

# AWSPowerShell- Une version mono-module pour Windows PowerShell



Cette version de AWS Tools for PowerShell est compatible et ne peut être installée que sur les ordinateurs Windows exécutant les PowerShell versions 2.0 à 5.1 de Windows. Il n'est pas compatible avec PowerShell Core 6.0 ou version ultérieure, ni avec aucun autre système d'exploitation (Linux ou macOS). Cette version consiste en un seul module volumineux qui prend en charge tous les AWS services.

Tout au long de ce guide, lorsque nous devons spécifier cette version uniquement, nous la désignons par son nom de module : AWSPowerShell.

## Comment utiliser ce guide

Le guide est divisé en plusieurs sections principales :

Installation du AWS Tools for PowerShell

Cette section explique comment installer le AWS Tools for PowerShell. Il explique comment vous inscrire AWS si vous n'avez pas encore de compte et comment créer un IAM utilisateur que vous pouvez utiliser pour exécuter les applets de commande.

Premiers pas avec AWS Tools for Windows PowerShell

Cette section décrit les principes fondamentaux de l'utilisation de AWS Tools for PowerShell, tels que la spécification des informations d'identification et des AWS régions, la recherche d'applets de commande pour un service particulier et l'utilisation d'alias pour les applets de commande.

Travaillez avec AWS les services du AWS Tools for PowerShell

Cette section contient des informations sur l'utilisation du AWS Tools for PowerShell pour effectuer certaines des AWS tâches les plus courantes.

## Sujets supplémentaires dans cette section

- Quoi de neuf dans le AWS Tools for PowerShell

# Quoi de neuf dans le AWS Tools for PowerShell

Pour obtenir des informations de haut niveau sur les nouveaux développements liés à la AWS Tools for PowerShell, consultez la page du produit à l'adresse https://aws.amazon.com/powershell/.

Voici les nouveautés des outils pour PowerShell.

13 septembre 2024 : version préliminaire pour l'observabilité

> Il s'agit de la documentation d'avant-première d'une fonctionnalité en version préliminaire. Elle est susceptible d’être modifiée.

L'observabilité est la mesure dans laquelle l'état actuel d'un système peut être déduit des données qu'il émet. L'observabilité a été ajoutée aux outils pour PowerShell, notamment la mise en œuvre d'un fournisseur de télémétrie.

# Installation du AWS Tools for PowerShell

Pour installer et utiliser les applets de commande AWS Tools for PowerShell, reportez-vous aux étapes décrites dans les rubriques suivantes.

Rubriques



## Installation du AWS Tools for PowerShell sous Windows

Un ordinateur Windows peut exécuter n'importe laquelle des options du AWS Tools for PowerShell package :

- **`AWS.Tools`**- La version modulaire de. AWS Tools for PowerShell Chaque AWS service est pris en charge par son propre petit module individuel, avec des modules de support partagés `AWS.Tools.Common` et`AWS.Tools.Installer`.
- AWSPowerShell. NetCore- La version à module unique de. AWS Tools for PowerShell Tous les AWS services sont pris en charge par ce grand module unique.

  > **Note**
  >
  > Notez que le module unique est peut-être trop volumineux pour être utilisé avec des fonctions AWS Lambda. Utilisez plutôt la version modulaire présentée ci-dessus.

- AWSPowerShell- L'ancienne version à module unique spécifique à Windows de. AWS Tools for PowerShell Tous les AWS services sont pris en charge par ce grand module unique.

Le package que vous choisissez dépend de la version et de l'édition de Windows que vous exécutez.

> **Note**
>
> Les outils pour Windows PowerShell (AWSPowerShellmodule) sont installés par défaut sur toutes les Amazon Machine Images (AMIs) basées sur Windows.

La configuration AWS Tools for PowerShell implique les tâches de haut niveau suivantes, décrites en détail dans cette rubrique.

1. Installez l'option de AWS Tools for PowerShell package adaptée à votre environnement.
2. Vérifiez que l'exécution de script est activée en exécutant l'applet de commande `Get-ExecutionPolicy`.
3. Importez le AWS Tools for PowerShell module dans votre PowerShell session.

## Prérequis

Les versions les plus récentes de PowerShell, y compris PowerShell Core, sont disponibles en téléchargement auprès de Microsoft à l'adresse Installation de différentes versions de PowerShell sur le site Web de Microsoft.

## Installer **`AWS.Tools`** sous Windows.

Vous pouvez installer la version modulaire de AWS Tools for PowerShell sur les ordinateurs qui exécutent Windows PowerShell 5.1, PowerShell Core 6.0 ou version ultérieure. Pour plus d'informations sur l'installation de PowerShell Core, voir Installation de différentes versions de PowerShell sur le site Web de Microsoft.

Vous pouvez installer `AWS.Tools` de l'une des trois façons suivantes :

- Utilisez les applets de commande du module `AWS.Tools.Installer`. Ce module simplifie l'installation et la mise à jour des autres `AWS.Tools` modules. `AWS.Tools.Installer`nécessite`PowerShellGet`, télécharge et installe automatiquement une version mise à jour de celui-ci. `AWS.Tools.Installer`synchronise automatiquement les versions de vos modules. Lorsque vous installez ou mettez à jour une version plus récente d'un module, les applets de commande intégrés mettent `AWS.Tools.Installer` automatiquement à jour tous les autres `AWS.Tools` modules vers la même version.

  Cette méthode est décrite dans la procédure qui suit.
- Téléchargez les modules depuis `AWS.Tools.zip` et insérez-les dans l'un des dossiers du module. Vous pouvez découvrir les dossiers de vos modules en affichant la valeur de la variable d'environnement `PSModulePath`.

> **Warning**
>
> Après avoir téléchargé le ZIP fichier et avant d'en extraire le contenu, vous devrez peut-être le débloquer. Cela se fait généralement en ouvrant les propriétés du fichier, en consultant l'onglet Général et en cochant la case Débloquer s'il en existe une.
> Si le ZIP fichier doit être débloqué mais que vous ne le faites pas, des erreurs similaires au message suivant peuvent s'afficher : « Module d'importation : Impossible de charger le fichier ou l'assemblage ».

- Installation de chaque module de service depuis la PowerShell galerie à l'aide de l'`Install-Module`applet de commande.

Pour installer sous **`AWS.Tools`** Windows à l'aide du **`AWS.Tools.Installer`** module

1. Démarrez une PowerShell session.

   > **Note**
   >
   > Nous vous recommandons de ne pas vous présenter PowerShell en tant qu'administrateur avec des autorisations élevées, sauf lorsque la tâche en cours l'exige. Ceci est dû au risque potentiel pour la sécurité et est incompatible avec le principe du moindre privilège.

2. Pour installer le package `AWS.Tools` modulaire, exécutez la commande suivante.

   ```
   PS > Install-Module -Name AWS.Tools.Installer

   Untrusted repository
   You are installing the modules from an untrusted repository. If you trust this
    repository, change its InstallationPolicy value by running the Set-PSRepository
    cmdlet. Are you sure
    you want to install the modules from 'PSGallery'?
   [Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
    "N"): y
   ```

   Si vous êtes averti que le référentiel n'est pas approuvé, le système vous demandera si vous souhaitez effectuer l'installation malgré tout. Entrez **y** pour autoriser PowerShell l'installation

du module. Pour éviter l'invite et installer le module sans approuver le référentiel, vous pourrez exécuter la commande avec le paramètre `-Force`.

```
PS > Install-Module -Name AWS.Tools.Installer -Force
```

3. Vous pouvez désormais installer le module pour chaque AWS service que vous souhaitez utiliser à l'aide de l'`Install-AWSToolsModule`applet de commande. Par exemple, la commande suivante installe les modules Amazon EC2 et Amazon S3. Cette commande installe également tous les modules dépendants nécessaires au fonctionnement du module spécifié. Par exemple, lorsque vous installez votre premier module de service `AWS.Tools`, il installe également `AWS.Tools.Common`. Il s'agit d'un module partagé requis par tous les modules AWS de service. Il supprime également les anciennes versions des modules et met à jour les autres modules avec la même version plus récente.

```
PS > Install-AWSToolsModule AWS.Tools.EC2,AWS.Tools.S3 -CleanUp
  Confirm
  Are you sure you want to perform this action?
  Performing the operation "Install-AWSToolsModule" on target "AWS Tools version
 4.0.0.0".
  [Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):

  Installing module AWS.Tools.Common version 4.0.0.0
  Installing module AWS.Tools.EC2 version 4.0.0.0
  Installing module AWS.Tools.Glacier version 4.0.0.0
  Installing module AWS.Tools.S3 version 4.0.0.0

  Uninstalling AWS.Tools version 3.3.618.0
  Uninstalling module AWS.Tools.Glacier
  Uninstalling module AWS.Tools.S3
  Uninstalling module AWS.Tools.SimpleNotificationService
  Uninstalling module AWS.Tools.SQS
  Uninstalling module AWS.Tools.Common
```

> **Note**
>
> L'`Install-AWSToolsModule`applet de commande télécharge tous les modules demandés depuis le `PSRepository` nom `PSGallery` (https://www.powershellgallery.com/) et considère qu'il s'agit d'une source fiable. Utilisez la

commande `Get-PSRepository -Name PSGallery` pour plus d'informations sur ce `PSRepository`.

Par défaut, la commande précédente installe les modules dans le dossier `%USERPROFILE%\Documents\WindowsPowerShell\Modules`. Pour installer le AWS Tools for PowerShell pour tous les utilisateurs d'un ordinateur, vous devez exécuter la commande suivante dans une PowerShell session que vous avez démarrée en tant qu'administrateur. Par exemple, la commande suivante installe le IAM module `%ProgramFiles%\WindowsPowerShell\Modules` dans le dossier accessible à tous les utilisateurs.

```
PS > Install-AWSToolsModule AWS.Tools.IdentityManagement -Scope AllUsers
```

Pour installer d'autres modules, exécutez des commandes similaires avec les noms de module appropriés, comme indiqué dans la PowerShell galerie.

## Installez AWSPowerShell. NetCore sous Windows

Vous pouvez installer le AWSPowerShell. NetCore sur les ordinateurs qui exécutent Windows avec les PowerShell versions 3 à 5.1, ou PowerShell Core 6.0 ou version ultérieure. Pour plus d'informations sur l'installation de PowerShell Core, consultez la section Installation de différentes versions de PowerShell sur le PowerShell site Web de Microsoft.

Vous pouvez installer AWSPowerShell. NetCore de deux manières

- Téléchargement du module depuis AWSPowerShell. NetCore.zip et en l'extrayant dans l'un des répertoires du module. Vous pouvez découvrir les répertoires de vos modules en affichant la valeur de la variable d'environnement `PSModulePath`.

> **Warning**
>
> Après avoir téléchargé le ZIP fichier et avant d'en extraire le contenu, vous devrez peut-être le débloquer. Cela se fait généralement en ouvrant les propriétés du fichier, en consultant l'onglet Général et en cochant la case Débloquer s'il en existe une.
> Si le ZIP fichier doit être débloqué mais que vous ne le faites pas, des erreurs similaires au message suivant peuvent s'afficher : « Module d'importation : Impossible de charger le fichier ou l'assemblage ».

- Installation depuis la PowerShell galerie à l'aide de l'`Install-Module`applet de commande, comme décrit dans la procédure suivante.

À installer AWSPowerShell. NetCore depuis la PowerShell galerie à l'aide de l'applet de commande Install-Module

Pour installer le AWSPowerShell. NetCore depuis la PowerShell Galerie, votre ordinateur doit exécuter la PowerShell version 5.0 ou ultérieure, ou la version PowerShell 3 ou une version ultérieure. PowerShellGet Exécutez la commande suivante.

```
PS > Install-Module -name AWSPowerShell.NetCore
```

Si vous exécutez en PowerShell tant qu'administrateur, la commande précédente s'installe AWS Tools for PowerShell pour tous les utilisateurs de l'ordinateur. Si vous vous lancez PowerShell en tant qu'utilisateur standard sans autorisation d'administrateur, cette même commande s'installe uniquement AWS Tools for PowerShell pour l'utilisateur actuel.

Pour effectuer une installation uniquement pour l'utilisateur actuel lorsque ce dernier dispose d'autorisations d'administrateur, exécutez la commande avec le jeu de paramètres `-Scope CurrentUser`, comme suit.

```
PS > Install-Module -name AWSPowerShell.NetCore -Scope CurrentUser
```

Bien que les versions PowerShell 3.0 et ultérieures chargent généralement des modules dans votre PowerShell session la première fois que vous exécutez une applet de commande dans le module, le. AWSPowerShell NetCore le module est trop volumineux pour prendre en charge cette fonctionnalité. Vous devez plutôt charger explicitement le AWSPowerShell. NetCore Intégrez le module principal à votre PowerShell session en exécutant la commande suivante.

```
PS > Import-Module AWSPowerShell.NetCore
```

Pour charger le AWSPowerShell. NetCore entrez automatiquement dans une PowerShell session, ajoutez cette commande à votre PowerShell profil. Pour plus d'informations sur la modification de votre PowerShell profil, consultez la section À propos des profils dans la PowerShell documentation.

## Installation AWSPowerShell sous Windows PowerShell

Vous pouvez l' AWS Tools for Windows PowerShell installer de deux manières :

- Télécharger le module depuis un AWSPowerShellfichier .zip et l'extraire dans l'un des répertoires du module. Vous pouvez découvrir les répertoires de vos modules en affichant la valeur de la variable d'environnement `PSModulePath`.

  > **Warning**
  >
  > Après avoir téléchargé le ZIP fichier et avant d'en extraire le contenu, vous devrez peut-être le débloquer. Cela se fait généralement en ouvrant les propriétés du fichier, en consultant l'onglet Général et en cochant la case Débloquer s'il en existe une.
  > Si le ZIP fichier doit être débloqué mais que vous ne le faites pas, des erreurs similaires au message suivant peuvent s'afficher : « Module d'importation : Impossible de charger le fichier ou l'assemblage ».

- Installation depuis la PowerShell galerie à l'aide de l'`Install-Module`applet de commande, comme décrit dans la procédure suivante.

Pour effectuer une installation AWSPowerShell depuis la PowerShell galerie à l'aide de l'applet de commande Install-Module

Vous pouvez l'installer AWSPowerShell depuis la PowerShell galerie si vous utilisez la PowerShell version 5.0 ou une version ultérieure, ou si vous avez installé PowerShellGetla version PowerShell 3 ou une version ultérieure. Vous pouvez installer et mettre à jour à AWSPowerShell partir de la PowerShellgalerie Microsoft en exécutant la commande suivante.

```
PS > Install-Module -Name AWSPowerShell
```

Pour charger automatiquement le AWSPowerShell module dans une PowerShell session, ajoutez l'`import-module`applet de commande précédente à votre PowerShell profil. Pour plus d'informations sur la modification de votre PowerShell profil, consultez la section À propos des profils dans la PowerShell documentation.

> **Note**
>
> Les outils pour Windows PowerShell sont installés par défaut sur toutes les Amazon Machine Images (AMIs) basées sur Windows.

# Activation de l'exécution du script

Pour charger les AWS Tools for PowerShell modules, vous devez activer l'exécution de PowerShell scripts. Pour activer l'exécution du script, exécutez l'applet de commande `Set-ExecutionPolicy` et définissez une stratégie `RemoteSigned`. Pour plus d'informations, consultez About Execution Policies (À propos des politiques d'exécution) sur le site Web Microsoft Technet.

> **Note**
>
> Ceci est une condition requise uniquement pour les ordinateurs qui exécutent Windows. La restriction de sécurité `ExecutionPolicy` n'est pas présente sur les autres systèmes d'exploitation.

Pour activer l'exécution du script

1. Les droits d'administrateurs sont requis pour définir la politique d'exécution. Si vous n'êtes pas connecté en tant qu'utilisateur disposant de droits d'administrateur, ouvrez une PowerShell session en tant qu'administrateur. Choisissez Démarrer, puis Tous les programmes. Choisissez Accessoires, puis Windows PowerShell. Cliquez avec le bouton droit sur Windows PowerShell, puis dans le menu contextuel, sélectionnez Exécuter en tant qu'administrateur.
2. À partir de l'invite de commande, entrez la commande suivante :

```
PS > Set-ExecutionPolicy RemoteSigned
```

> **Note**
>
> Sur un système 64 bits, vous devez effectuer cette opération séparément pour la version 32 bits de PowerShell Windows PowerShell (x86).

Si la politique d'exécution n'est pas correctement définie, PowerShell affiche l'erreur suivante chaque fois que vous essayez d'exécuter un script, tel que votre profil.

```
File C:\Users\username\Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1
 cannot be loaded because the execution
 of scripts is disabled on this system. Please see "get-help about_signing" for more
 details.
```

```
At line:1 char:2
+ . <<<<  'C:\Users\username\Documents\WindowsPowerShell
\Microsoft.PowerShell_profile.ps1'
    + CategoryInfo          : NotSpecified: (:) [], PSSecurityException
    + FullyQualifiedErrorId : RuntimeException
```

Le programme d' PowerShell installation d'Outils pour Windows met automatiquement PSModulePathà jour le pour inclure l'emplacement du répertoire contenant le `AWSPowerShell` module.

Comme il `PSModulePath` inclut l'emplacement du répertoire du AWS module, l'`Get-Module -ListAvailable`applet de commande affiche le module.

```
PS > Get-Module -ListAvailable

ModuleType Name                      ExportedCommands
---------- ----                      ----------------
Manifest   AppLocker                 {}
Manifest   BitsTransfer              {}
Manifest   PSDiagnostics             {}
Manifest   TroubleshootingPack       {}
Manifest   AWSPowerShell             {Update-EBApplicationVersion, Set-DPStatus,
 Remove-IAMGroupPol...
```

## Gestion des versions

AWS publie AWS Tools for PowerShell régulièrement de nouvelles versions pour prendre en charge les nouveaux AWS services et fonctionnalités. Pour déterminer la version des outils que vous avez installée, exécutez l'AWSPowerShellVersionapplet de commande Get-.

```
PS > Get-AWSPowerShellVersion

Tools for PowerShell
Version 4.1.11.0
Copyright 2012-2021 Amazon.com, Inc. or its affiliates. All Rights Reserved.

Amazon Web Services SDK for .NET
Core Runtime Version 3.7.0.12
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

Release notes: https://github.com/aws/aws-tools-for-powershell/blob/master/CHANGELOG.md
```

```
This software includes third party software subject to the following copyrights:
- Logging from log4net, Apache License
[http://logging.apache.org/log4net/license.html]
```

Vous pouvez également ajouter le `-ListServiceVersionInfo` paramètre à une AWSPowerShellVersion commande Get- pour voir la liste des AWS services pris en charge dans la version actuelle des outils. Si vous utilisez l'option `AWS.Tools.*` modulaire, seuls les modules que vous avez importés seront affichés.

```
PS > Get-AWSPowerShellVersion -ListServiceVersionInfo
...

Service                            Noun Prefix Module Name                        SDK

 Assembly

 Version
-------                            ----------- -----------
 -----------
Alexa For Business                 ALXB        AWS.Tools.AlexaForBusiness
 3.7.0.11
Amplify Backend                    AMPB        AWS.Tools.AmplifyBackend
 3.7.0.11
Amazon API Gateway                 AG          AWS.Tools.APIGateway
 3.7.0.11
Amazon API Gateway Management API  AGM         AWS.Tools.ApiGatewayManagementApi
 3.7.0.11
Amazon API Gateway V2              AG2         AWS.Tools.ApiGatewayV2
 3.7.0.11
Amazon Appflow                     AF          AWS.Tools.Appflow
 3.7.1.4
Amazon Route 53                    R53         AWS.Tools.Route53
 3.7.0.12
Amazon Route 53 Domains            R53D        AWS.Tools.Route53Domains
 3.7.0.11
Amazon Route 53 Resolver           R53R        AWS.Tools.Route53Resolver
 3.7.1.5
Amazon Simple Storage Service (S3) S3          AWS.Tools.S3
 3.7.0.13
...
```

Pour déterminer la version PowerShell que vous utilisez, entrez `$PSVersionTable` pour afficher le contenu de la variable PSVersionTable automatique $.

```
PS > $PSVersionTable

Name                           Value
----                           -----
PSVersion                      6.2.2
PSEdition                      Core
GitCommitId                    6.2.2
OS                             Darwin 18.7.0 Darwin Kernel Version 18.7.0: Tue Aug 20
 16:57:14 PDT 2019; root:xnu-4903.271.2~2/RELEASE_X86_64
Platform                       Unix
PSCompatibleVersions           {1.0, 2.0, 3.0, 4.0…}
PSRemotingProtocolVersion      2.3
SerializationVersion           1.1.0.1
WSManStackVersion              3.0
```

## Mettre à jour AWS Tools for PowerShell le sous Windows

Régulièrement, au fur et à mesure que des versions mises à jour AWS Tools for PowerShell sont publiées, vous devez mettre à jour la version que vous exécutez localement.

### Mettre à jour les modules modularisés **`AWS.Tools`**

Pour mettre à jour vos `AWS.Tools` modules avec la dernière version, exécutez la commande suivante :

```
PS > Update-AWSToolsModule -CleanUp
```

Cette commande met à jour tous les modules `AWS.Tools` actuellement installés et, après une mise à jour réussie, supprime les autres versions installées.

> **Note**
>
> L'`Update-AWSToolsModule`applet de commande télécharge tous les modules depuis le `PSRepository` nom `PSGallery` (https://www.powershellgallery.com/) et considère qu'il s'agit d'une source fiable. Utilisez la commande `Get-PSRepository -Name PSGallery` pour plus d'informations sur ce `PSRepository`.

## Mettre à jour les outils pour PowerShell Core

Exécutez l'`Get-AWSPowerShellVersion`applet de commande pour déterminer la version que vous exécutez et comparez-la à la version de Tools for Windows disponible sur le PowerShell site Web de la PowerShell galerie. Nous vous suggérons d'effectuer cette vérification toutes les deux à trois semaines. Support pour les nouvelles commandes et AWS services n'est disponible qu'après la mise à jour vers une version compatible avec ce support.

Avant d'installer une version plus récente de AWSPowerShell. NetCore, désinstallez le module existant. Fermez toutes les PowerShell sessions ouvertes avant de désinstaller le package existant. Exécutez la commande suivante pour désinstaller le package.

```
PS > Uninstall-Module -Name AWSPowerShell.NetCore -AllVersions
```

Une fois le package désinstallé, installez le module mis à jour en exécutant la commande suivante.

```
PS > Install-Module -Name AWSPowerShell.NetCore
```

Après l'installation, exécutez la commande `Import-Module AWSPowerShell.NetCore` pour charger les applets de commande mis à jour dans votre PowerShell session.

## Mettre à jour les outils pour Windows PowerShell

Exécutez l'`Get-AWSPowerShellVersion`applet de commande pour déterminer la version que vous exécutez et comparez-la à la version de Tools for Windows disponible sur le PowerShell site Web de la PowerShell galerie. Nous vous suggérons d'effectuer cette vérification toutes les deux à trois semaines. Support pour les nouvelles commandes et AWS services n'est disponible qu'après la mise à jour vers une version compatible avec ce support.

- Si vous avez effectué l'installation à l'aide de l'applet de commande `Install-Module`, exécutez les commandes suivantes.

  ```
  PS > Uninstall-Module -Name AWSPowerShell -AllVersions
  PS > Install-Module -Name AWSPowerShell
  ```

- Si vous avez effectué l'installation à l'aide d'un ZIP fichier téléchargé :

  1. Téléchargez la version la plus récente sur le site PowerShell Web Tools for. Comparez le numéro de version du package dans le nom de fichier téléchargé avec le numéro de version que vous obtenez lorsque vous exécutez l'applet de commande `Get-AWSPowerShellVersion`.

2. Si le nombre de versions téléchargées est supérieur à celui de la version que vous avez installée, fermez tous les outils pour PowerShell consoles Windows.
3. Installez la version la plus récente des outils pour Windows PowerShell.

Après l'installation, exécutez `Import-Module AWSPowerShell` pour charger les applets de commande mis à jour dans votre PowerShell session. Vous pouvez également exécuter la AWS Tools for PowerShell console personnalisée depuis le menu Démarrer.

# Installation AWS Tools for PowerShell sous Linux ou macOS

Cette rubrique fournit des instructions sur la façon d'installer AWS Tools for PowerShell le sous Linux ou macOS.

## Présentation de la configuration

Pour l'installer AWS Tools for PowerShell sur un ordinateur Linux ou macOS, vous pouvez choisir entre deux options de package :

- **`AWS.Tools`**— La version modulaire de. AWS Tools for PowerShell Chaque AWS service est pris en charge par son propre petit module individuel, avec des modules de support partagés`AWS.Tools.Common`.
- AWSPowerShell. NetCore— La version à module unique de grande taille de. AWS Tools for PowerShell Tous les AWS services sont pris en charge par ce grand module unique.

> **Note**
>
> Notez que le module unique est peut-être trop volumineux pour être utilisé avec des fonctions AWS Lambda. Utilisez plutôt la version modulaire présentée ci-dessus.

La configuration de l'une ou l'autre de ces options sur un ordinateur exécutant Linux ou macOS implique les tâches suivantes, décrites en détail plus loin dans cette rubrique :

1. Installez PowerShell Core 6.0 ou version ultérieure sur un système compatible.
2. Après avoir installé PowerShell Core, commencez PowerShell par exécuter `pwsh` dans le shell de votre système.
3. Installez l'un `AWS.Tools` ou l'autre AWSPowerShell. NetCore.

4. Exécutez l'`Import-Module`applet de commande appropriée pour importer le module dans votre PowerShell session.
5. Exécutez l'AWSDefaultConfigurationapplet de commande Initialize- pour fournir vos informations d'identification. AWS

## Prérequis

Pour exécuter le AWS Tools for PowerShell Core, votre ordinateur doit exécuter PowerShell Core 6.0 ou une version ultérieure.

- Pour obtenir la liste des versions de plate-forme Linux prises en charge et pour savoir comment installer la dernière version de PowerShell sur un ordinateur Linux, consultez la section Installation sous Linux PowerShell sur le site Web de Microsoft. Certains systèmes d'exploitation basés sur Linux, comme Arch, Kali et Raspbian, ne sont pas officiellement pris en charge, mais offrent différents niveaux de support communautaire.
- Pour plus d'informations sur les versions de macOS prises en charge et sur la façon d'installer la dernière version de PowerShell macOS, consultez la section Installation sous macOS PowerShell sur le site Web de Microsoft.

## Installer **`AWS.Tools`** sur Linux ou sur macOS

Vous pouvez installer la version modulaire de AWS Tools for PowerShell sur les ordinateurs qui exécutent PowerShell Core 6.0 ou version ultérieure. Pour plus d'informations sur l'installation de PowerShell Core, consultez la section Installation de différentes versions de PowerShell sur le PowerShell site Web de Microsoft.

Vous pouvez installer `AWS.Tools` de l'une des trois façons suivantes :

- Utilisez les applets de commande du module `AWS.Tools.Installer`. Ce module simplifie l'installation et la mise à jour des autres `AWS.Tools` modules. `AWS.Tools.Installer`nécessite`PowerShellGet`, télécharge et installe automatiquement une version mise à jour de celui-ci. `AWS.Tools.Installer`synchronise automatiquement les versions de vos modules. Lorsque vous installez ou mettez à jour une version plus récente d'un module, les applets de commande intégrés mettent `AWS.Tools.Installer` automatiquement à jour tous les autres `AWS.Tools` modules vers la même version.

  Cette méthode est décrite dans la procédure qui suit.

- Téléchargez les modules depuis `AWS.Tools.zip` et ajoutez-les dans l'un des répertoires du module. Vous pouvez découvrir les répertoires de vos modules en imprimant la valeur de la variable `$Env:PSModulePath`.
- Installation de chaque module de service depuis la PowerShell galerie à l'aide de l'`Install-Module`applet de commande.

Pour installer sous **`AWS.Tools`** Linux ou macOS à l'aide du **`AWS.Tools.Installer`** module

1. Démarrez une session PowerShell Core en exécutant la commande suivante.

```
$ pwsh
```

> **Note**
>
> Nous vous recommandons de ne pas vous présenter PowerShell en tant qu'administrateur avec des autorisations élevées, sauf lorsque la tâche en cours l'exige. Ceci est dû au risque potentiel pour la sécurité et est incompatible avec le principe du moindre privilège.

2. Pour installer le package `AWS.Tools` modularisé à l'aide du module `AWS.Tools.Installer`, exécutez la commande suivante.

```
PS > Install-Module -Name AWS.Tools.Installer

Untrusted repository
You are installing the modules from an untrusted repository. If you trust this
 repository, change its InstallationPolicy value by running the Set-PSRepository
 cmdlet. Are you sure
 you want to install the modules from 'PSGallery'?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "N"): y
```

Si vous êtes averti que le référentiel n'est pas approuvé, le système vous demande si vous souhaitez effectuer l'installation malgré tout. Entrez **y** pour autoriser PowerShell l'installation du module. Pour éviter l'invite et installer le module sans approuver le référentiel, vous pouvez exécuter la commande suivante.

```
PS > Install-Module -Name AWS.Tools.Installer -Force
```

3. Vous pouvez maintenant installer le module pour chaque service que vous souhaitez utiliser. Par exemple, la commande suivante installe les modules Amazon EC2 et Amazon S3. Cette commande installe également tous les modules dépendants nécessaires au fonctionnement du module spécifié. Par exemple, lorsque vous installez votre premier module de service `AWS.Tools`, il installe également `AWS.Tools.Common`. Il s'agit d'un module partagé requis par tous les modules AWS de service. Il supprime également les anciennes versions des modules et met à jour les autres modules avec la même version plus récente.

```
PS > Install-AWSToolsModule AWS.Tools.EC2,AWS.Tools.S3 -CleanUp
Confirm
Are you sure you want to perform this action?
  Performing the operation "Install-AWSToolsModule" on target "AWS Tools version
 4.0.0.0".
  [Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):

  Installing module AWS.Tools.Common version 4.0.0.0
  Installing module AWS.Tools.EC2 version 4.0.0.0
  Installing module AWS.Tools.Glacier version 4.0.0.0
  Installing module AWS.Tools.S3 version 4.0.0.0

  Uninstalling AWS.Tools version 3.3.618.0
  Uninstalling module AWS.Tools.Glacier
  Uninstalling module AWS.Tools.S3
  Uninstalling module AWS.Tools.SimpleNotificationService
  Uninstalling module AWS.Tools.SQS
  Uninstalling module AWS.Tools.Common
```

> **Note**
>
> L'applet de commande `Install-AWSToolsModule` télécharge tous les modules demandés à partir du `PSRepository` appelé `PSGallery` (https://www.powershellgallery.com/) et considère le référentiel comme une source fiable. Utilisez la commande `Get-PSRepository -Name PSGallery` pour plus d'informations sur ce `PSRepository`.

La commande précédente installe les modules dans les répertoires par défaut de votre système. Les répertoires réels dépendent de la distribution et de la version de votre système d'exploitation ainsi PowerShell que de la version que vous avez installée. Par exemple, si vous avez installé PowerShell 7 sur un système de type RHEL, les modules par défaut se trouvent probablement dans `/opt/microsoft/powershell/7/Modules` (ou`$PSHOME/Modules`) et les modules utilisateur se trouvent probablement dans. `~/.local/share/powershell/Modules` Pour plus d'informations, consultez la section Installer sous Linux PowerShell sur le PowerShell site Web de Microsoft. Pour voir où les modules sont installés, exécutez la commande suivante :

```
PS > Get-Module -ListAvailable
```

Pour installer d'autres modules, exécutez des commandes similaires avec les noms de module appropriés, comme indiqué dans la PowerShell galerie.

## Installez AWSPowerShell. NetCore sous Linux ou macOS

Pour effectuer une mise à niveau vers une version plus récente de AWSPowerShell. NetCore, suivez les instructions figurant dansMettre à jour AWS Tools for PowerShell le sous Linux ou macOS. Désinstallez les versions antérieures de AWSPowerShell. NetCore premier.

Vous pouvez installer AWSPowerShell. NetCore de l'une des deux manières suivantes :

- Téléchargez le module depuis `AWSPowerShell.NetCore.zip` et ajoutez-le dans l'un des répertoires du module. Vous pouvez découvrir les répertoires de vos modules en imprimant la valeur de la variable `$Env:PSModulePath`.
- Installation depuis la PowerShell galerie à l'aide de l'`Install-Module`applet de commande, comme décrit dans la procédure suivante.

### À installer AWSPowerShell. NetCore sous Linux ou macOS à l'aide de l'applet de commande Install-Module

Démarrez une session PowerShell Core en exécutant la commande suivante.

```
$ pwsh
```

> **Note**
>
> Nous vous recommandons de ne pas commencer PowerShell par courir `sudo pwsh` pour exécuter PowerShell avec des droits d'administrateur élevés. Ceci est dû au risque potentiel pour la sécurité et est incompatible avec le principe du moindre privilège.

Pour installer le AWSPowerShell. NetCore package mono-module depuis la PowerShell galerie, exécutez la commande suivante.

```
PS > Install-Module -Name AWSPowerShell.NetCore

Untrusted repository
You are installing the modules from an untrusted repository. If you trust this
 repository, change its InstallationPolicy value by running the Set-PSRepository
 cmdlet. Are you sure
 you want to install the modules from 'PSGallery'?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "N"): y
```

Si vous êtes averti que le référentiel n'est pas approuvé, le système vous demande si vous souhaitez effectuer l'installation malgré tout. Entrez **y** pour autoriser PowerShell l'installation du module. Pour éviter l'invite sans approuver le référentiel, vous pouvez exécuter la commande suivante.

```
PS > Install-Module -Name AWSPowerShell.NetCore -Force
```

Il n'est pas nécessaire d'exécuter cette commande en tant que root, sauf si vous souhaitez l'installer AWS Tools for PowerShell pour tous les utilisateurs d'un ordinateur. Pour ce faire, exécutez la commande suivante dans une PowerShell session que vous avez démarrée`sudo pwsh`.

```
PS > Install-Module -Scope AllUsers -Name AWSPowerShell.NetCore -Force
```

## Exécution de script

La commande `Set-ExecutionPolicy` n'est pas disponible sur les systèmes autres que Windows. Vous pouvez exécuter`Get-ExecutionPolicy`, ce qui montre que le paramètre de politique d'exécution par défaut dans PowerShell Core exécuté sur des systèmes autres que Windows est`Unrestricted`. Pour plus d'informations, consultez About Execution Policies (À propos des politiques d'exécution) sur le site Web Microsoft Technet.

Comme il `PSModulePath` inclut l'emplacement du répertoire du AWS module, l'`Get-Module -ListAvailable`applet de commande affiche le module que vous avez installé.

**`AWS.Tools`**

```
PS > Get-Module -ListAvailable

    Directory: /Users/username/.local/share/powershell/Modules

ModuleType Version    Name                                PSEdition ExportedCommands
---------- -------    ----                                --------- ----------------
Binary     3.3.563.1  AWS.Tools.Common                    Desk      {Clear-
AWSHistory, Set-AWSHistoryConfiguration, Initialize-AWSDefaultConfiguration, Clear-
AWSDefaultConfigurat…
```

AWSPowerShell.NetCore

```
PS > Get-Module -ListAvailable

Directory: /Users/username/.local/share/powershell/Modules

ModuleType Version    Name                                ExportedCommands
---------- -------    ----                                ----------------
Binary     3.3.563.1  AWSPowerShell.NetCore
```

## Configurez une PowerShell console pour utiliser le AWS Tools for PowerShell Core (AWSPowerShell. NetCore Uniquement)

PowerShell Core charge généralement les modules automatiquement chaque fois que vous exécutez une applet de commande dans le module. Mais cela ne fonctionne pas pour AWSPowerShell. NetCore en raison de sa grande taille. Pour commencer à courir AWSPowerShell. NetCore applets de commande, vous devez d'abord exécuter la `Import-Module AWSPowerShell.NetCore` commande. Ceci n'est pas requis pour les applets de commande dans les modules `AWS.Tools`.

## Initialisez votre session PowerShell

Lorsque vous démarrez PowerShell sur un système Linux ou macOS après avoir installé le AWS Tools for PowerShell, vous devez exécuter Initialize- AWSDefaultConfiguration pour spécifier la clé d'accès à utiliser. AWS Pour plus d'informations sur `Initialize-AWSDefaultConfiguration`, consultez Utilisation des informations d'identification AWS.

> **Note**
>
> Dans les versions antérieures (antérieures à la version 3.3.96.0) de AWS Tools for PowerShell, cette applet de commande était nommée. `Initialize-AWSDefaults`

## Gestion des versions

AWS publie AWS Tools for PowerShell régulièrement de nouvelles versions pour prendre en charge les nouveaux AWS services et fonctionnalités. Pour déterminer la version AWS Tools for PowerShell que vous avez installée, exécutez l'AWSPowerShellVersionapplet de commande Get-.

```
PS > Get-AWSPowerShellVersion

Tools for PowerShell
Version 4.0.123.0
Copyright 2012-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.

Amazon Web Services SDK for .NET
Core Runtime Version 3.3.103.22
Copyright 2009-2015 Amazon.com, Inc. or its affiliates. All Rights Reserved.

Release notes: https://github.com/aws/aws-tools-for-powershell/blob/master/CHANGELOG.md

This software includes third party software subject to the following copyrights:
- Logging from log4net, Apache License
[http://logging.apache.org/log4net/license.html]
```

Pour consulter la liste des AWS services pris en charge dans la version actuelle des outils, ajoutez le `-ListServiceVersionInfo` paramètre à une AWSPowerShellVersion applet de commande Get-.

Pour déterminer la version PowerShell que vous utilisez, entrez `$PSVersionTable` pour afficher le contenu de la variable `$PSVersionTable` automatique.

```
PS > $PSVersionTable
Name                           Value
----                           -----
PSVersion                      6.2.2
PSEdition                      Core
GitCommitId                    6.2.2
```

```
OS                              Darwin 18.7.0 Darwin Kernel Version 18.7.0: Tue Aug 20
 16:57:14 PDT 2019; root:xnu-4903.271.2~2/RELEASE_X86_64
Platform                        Unix
PSCompatibleVersions            {1.0, 2.0, 3.0, 4.0…}
PSRemotingProtocolVersion       2.3
SerializationVersion            1.1.0.1
WSManStackVersion               3.0
```

## Mettre à jour AWS Tools for PowerShell le sous Linux ou macOS

Régulièrement, à mesure que des versions mises à jour AWS Tools for PowerShell sont publiées, vous devez mettre à jour la version que vous utilisez localement.

### Mettre à jour les modules modularisés **`AWS.Tools`**

Pour mettre à jour vos `AWS.Tools` modules avec la dernière version, exécutez la commande suivante :

```
PS > Update-AWSToolsModule -CleanUp
```

Cette commande met à jour tous les modules `AWS.Tools` actuellement installés et, pour les modules qui ont été mis à jour avec succès, supprime les versions antérieures.

> **Note**
>
> L'applet de commande `Update-AWSToolsModule` télécharge tous les modules à partir du `PSRepository` appelé `PSGallery` (https://www.powershellgallery.com/) et considère le référentiel comme une source fiable. Utilisez la commande `Get-PSRepository -Name PSGallery` pour plus d'informations sur ce `PSRepository`.

### Mettre à jour les outils pour PowerShell Core

Exécutez l'`Get-AWSPowerShellVersion`applet de commande pour déterminer la version que vous exécutez et comparez-la à la version de Tools for Windows disponible sur le PowerShell site Web de la PowerShell galerie. Nous vous suggérons d'effectuer cette vérification toutes les deux à trois semaines. Support pour les nouvelles commandes et les nouveaux AWS services uniquement après la mise à jour vers une version avec ce support.

Avant d'installer une version plus récente de AWSPowerShell. NetCore, désinstallez le module existant. Fermez toutes les PowerShell sessions ouvertes avant de désinstaller le package existant. Exécutez la commande suivante pour désinstaller le package.

```
PS > Uninstall-Module -Name AWSPowerShell.NetCore -AllVersions
```

Une fois le package désinstallé, installez le module mis à jour en exécutant la commande suivante.

```
PS > Install-Module -Name AWSPowerShell.NetCore
```

Après l'installation, exécutez la commande `Import-Module AWSPowerShell.NetCore` pour charger les applets de commande mis à jour dans votre PowerShell session.

### Informations connexes

- Premiers pas avec AWS Tools for Windows PowerShell
- Travaillez avec AWS les services du AWS Tools for PowerShell

## Migration de la AWS Tools for PowerShell version 3.3 vers la version 4

AWS Tools for PowerShell la version 4 est une mise à jour rétrocompatible de la version 3.3. AWS Tools for PowerShell Elle apporte des améliorations significatives tout en maintenant le comportement existant de l'applet de commande.

Vos scripts existants devraient continuer à fonctionner après la mise à niveau vers la nouvelle version. Toutefois, nous vous recommandons de les tester soigneusement avant de procéder à la mise à niveau de vos environnements de production.

Cette section décrit les modifications et décrit l'impact qu'elles peuvent avoir sur vos scripts.

### Nouvelle version **`AWS.Tools`** entièrement modulaire

Le AWSPowerShell. NetCore et les AWSPowerShell colis étaient « monolithiques ». Cela signifiait que tous les AWS services étaient pris en charge dans le même module, ce qui le rendait très volumineux et augmentait à mesure que chaque nouveau AWS service ou fonctionnalité était

ajouté. Le nouveau `AWS.Tools` package est divisé en modules plus petits qui vous permettent de télécharger et d'installer uniquement ceux dont vous avez besoin pour les AWS services que vous utilisez. Le package comprend un module `AWS.Tools.Common` partagé requis par tous les autres modules, et un module `AWS.Tools.Installer` qui simplifie l'installation, la mise à jour et la suppression des modules selon les besoins.

Ceci permet également l'importation automatique des applets de commande lors du premier appel, sans devoir d'abord appeler `Import-module`. Cependant, pour interagir avec l'associé. NETobjets avant d'appeler une applet de commande, vous devez toujours appeler `Import-Module` pour PowerShell connaître les objets concernés. NETtypes.

Par exemple, la commande suivante comporte une référence à `Amazon.EC2.Model.Filter`. Ce type de référence ne peut pas déclencher l'importation automatique, vous devez donc d'abord appeler `Import-Module` pour éviter l'échec de la commande.

```
PS > $filter = [Amazon.EC2.Model.Filter]@{Name="vpc-id";Values="vpc-1234abcd"}
  InvalidOperation: Unable to find type [Amazon.EC2.Model.Filter].
```

```
PS > Import-Module AWS.Tools.EC2
PS > $filter = [Amazon.EC2.Model.Filter]@{Name="vpc-id";Values="vpc-1234abcd"}
PS > Get-EC2Instance -Filter $filter -Select Reservations.Instances.InstanceId
  i-0123456789abcdefg
  i-0123456789hijklmn
```

## Nouvelle applet de commande **`Get-AWSService`**

Pour vous aider à découvrir le nom des modules de chaque AWS service de la `AWS.Tools` collection de modules, vous pouvez utiliser l'`Get-AWSService`applet de commande.

```
PS > Get-AWSService
  Service : ACMPCA
  CmdletNounPrefix : PCA
  ModuleName : AWS.Tools.ACMPCA
  SDKAssemblyVersion : 3.3.101.56
  ServiceName : Certificate Manager Private Certificate Authority

  Service : AlexaForBusiness
  CmdletNounPrefix : ALXB
  ModuleName : AWS.Tools.AlexaForBusiness
  SDKAssemblyVersion : 3.3.106.26
```

```
  ServiceName : Alexa For Business
  ...
```

## Nouveau paramètre **`-Select`** pour contrôler l'objet renvoyé par une applet de commande

La plupart des applets de commande de la version 4 prennent en charge un nouveau paramètre `-Select`. Chaque applet de commande appelle le AWS service à votre place APIs à l'aide du. AWS SDK for .NET Le AWS Tools for PowerShell client convertit ensuite la réponse en un objet que vous pouvez utiliser dans vos PowerShell scripts et rediriger vers d'autres commandes. Parfois, l' PowerShell objet final comporte plus de champs ou de propriétés dans la réponse d'origine que ce dont vous avez besoin, et d'autres fois, vous souhaiterez peut-être que l'objet inclue des champs ou des propriétés de la réponse qui n'y figurent pas par défaut. Le `-Select` paramètre vous permet de spécifier ce qui est inclus dans le. NETobjet renvoyé par l'applet de commande.

Par exemple, l'Get-S3Objectapplet de commande appelle l'opération Amazon S3. SDK ListObjects Cette opération renvoie un ListObjectsResponseobjet. Toutefois, par défaut, l'`Get-S3Object`applet de commande renvoie uniquement l'`S3Objects`élément de SDK réponse à l' PowerShell utilisateur. Dans l'exemple suivant, cet objet est un tableau avec deux éléments.

```
PS > Get-S3Object -BucketName amzn-s3-demo-bucket

ETag : "01234567890123456789012345678901111"
BucketName : amzn-s3-demo-bucket
Key : file1.txt
LastModified : 9/30/2019 1:31:40 PM
Owner : Amazon.S3.Model.Owner
Size : 568
StorageClass : STANDARD

ETag : "01234567890123456789012345678902222"
BucketName : amzn-s3-demo-bucket
Key : file2.txt
LastModified : 7/15/2019 9:36:54 AM
Owner : Amazon.S3.Model.Owner
Size : 392
StorageClass : STANDARD
```

Dans AWS Tools for PowerShell la version 4, vous pouvez spécifier `-Select *` de renvoyer l'intégralité. NETobjet de réponse renvoyé par l'SDKAPIappel.

```
PS > Get-S3Object -BucketName amzn-s3-demo-bucket -Select *
  IsTruncated    : False
  NextMarker     :
  S3Objects      : {file1.txt, file2.txt}
  Name           : amzn-s3-demo-bucket
  Prefix         :
  MaxKeys        : 1000
  CommonPrefixes : {}
  Delimiter      :
```

Vous pouvez également spécifier le chemin d'accès vers la propriété imbriquée spécifique de votre choix. L'exemple suivant renvoie uniquement la propriété `Key` de chaque élément du tableau `S3Objects`.

```
PS > Get-S3Object -BucketName amzn-s3-demo-bucket -Select S3Objects.Key
file1.txt
file2.txt
```

Dans certaines situations, il peut être utile de renvoyer un paramètre d'applet de commande. Vous pouvez effectuer cette opération avec `-Select ^ParameterName`. Cette fonctionnalité supplante le paramètre `-PassThru`, qui est toujours disponible, mais obsolète.

```
PS > Get-S3Object -BucketName amzn-s3-demo-bucket -Select S3Objects.Key |
>> Write-S3ObjectTagSet -Select ^Key -BucketName amzn-s3-demo-bucket -Tagging_TagSet
 @{ Key='key'; Value='value'}
  file1.txt
  file2.txt
```

La rubrique de référence de chaque applet de commande identifie si elle prend en charge le paramètre `-Select`.

## Limitation plus cohérente du nombre d'éléments dans la sortie

Les versions antérieures de vous AWS Tools for PowerShell permettaient d'utiliser le `-MaxItems` paramètre pour spécifier le nombre maximum d'objets renvoyés dans le résultat final.

Ce comportement est supprimé de `AWS.Tools`.

Ce comportement est obsolète dans. AWSPowerShell NetCore et AWSPowerShell, et seront supprimés de ces versions dans une future version.

Si le service sous-jacent API prend en charge un `MaxItems` paramètre, celui-ci est toujours disponible et fonctionne comme API indiqué. Mais il ne présente plus le comportement qui consiste à limiter le nombre d'éléments renvoyés dans la sortie de l'applet de commande.

Pour limiter le nombre d'éléments renvoyés dans la sortie finale, dirigez la sortie vers l'`Select-Object`applet de commande et spécifiez le `-First` *`n`* paramètre, où *`n`* est le nombre maximum d'éléments à inclure dans le résultat final.

```
PS > Get-S3ObjectV2 -BucketName amzn-s3-demo-bucket -Select S3Objects.Key | select -
first 2
file1.txt
file2.txt
```

Tous les AWS services ne sont pas pris `-MaxItems` en charge de la même manière, ce qui élimine cette incohérence et les résultats inattendus qui se produisaient parfois. De plus, `-MaxItems` combiné avec le nouveau paramètre `-Select` entraînait parfois des résultats déroutants.

## Paramètres de flux plus faciles à utiliser

Les paramètres de type `Stream` ou `byte[]` peuvent désormais accepter les valeurs `string`, `string[]` ou `FileInfo`.

Par exemple, vous pouvez utiliser l'un des exemples suivants.

```
PS > Invoke-LMFunction -FunctionName MyTestFunction -PayloadStream '{
>> "some": "json"
>> }'
```

```
PS > Invoke-LMFunction -FunctionName MyTestFunction -PayloadStream (ls .\some.json)
```

```
PS > Invoke-LMFunction -FunctionName MyTestFunction -PayloadStream @('{', '"some":
 "json"', '}')
```

AWS Tools for PowerShell convertit toutes les chaînes en `byte[]` utilisant le codage UTF -8.

## Extension du pipeline par nom de propriété

Pour rendre l'expérience utilisateur plus cohérente, vous pouvez désormais transmettre l'entrée de pipeline en spécifiant le nom de propriété pour n'importe quel paramètre.

Dans l'exemple suivant, nous créons un objet personnalisé avec des propriétés dont les noms correspondent aux noms de paramètres de l'applet de commande cible. Lorsque l'applet de commande s'exécute, elle utilise automatiquement ces propriétés comme paramètres.

```
PS > [pscustomobject] @{ BucketName='amzn-s3-demo-bucket'; Key='file1.txt';
 PartNumber=1 } | Get-S3ObjectMetadata
```

> **Note**
>
> Certaines propriétés prenaient cela en charge dans les versions antérieures de AWS Tools for PowerShell. La version 4 rend cela plus cohérent en l'activant pour tous les paramètres.

## Paramètres communs statiques

Pour améliorer la cohérence dans la version 4.0 de AWS Tools for PowerShell, tous les paramètres sont statiques.

Dans les versions antérieures de AWS Tools for PowerShell, certains paramètres courants tels que`AccessKey`,`SecretKey`, ou `ProfileName`Region, étaient dynamiques, tandis que tous les autres paramètres étaient statiques. Cela peut créer des problèmes car PowerShell lie les paramètres statiques aux paramètres dynamiques. Par exemple, imaginons que vous ayez exécuté la commande suivante.

```
PS > Get-EC2Region -Region us-west-2
```

Les versions antérieures de PowerShell liaient la valeur `us-west-2` au paramètre `-RegionName` statique plutôt qu'au paramètre `-Region` dynamique. Ceci pouvait perturber les utilisateurs.

## **`AWS.Tools`** déclare et applique les paramètres obligatoires

Les modules `AWS.Tools.*` déclarent et appliquent désormais les paramètres obligatoires de l'applet de commande. Lorsqu'un AWS service déclare qu'un paramètre de API est obligatoire, il vous PowerShell demande le paramètre d'applet de commande correspondant si vous ne l'avez pas spécifié. Ceci s'applique uniquement à `AWS.Tools`. Pour garantir la rétrocompatibilité, cela ne s'applique pas à AWSPowerShell. NetCore ou AWSPowerShell.

## Tous les paramètres sont nullables

Vous pouvez désormais attribuer `$null` aux paramètres de type valeur (nombres et dates). Cette modification ne devrait pas affecter les scripts existants. Ceci vous permet de contourner l'invite pour un paramètre obligatoire. Les paramètres obligatoires sont appliqués dans `AWS.Tools` uniquement.

Si vous exécutez l'exemple suivant en utilisant la version 4, il contournera efficacement la validation côté client, car vous fournissez une « valeur » pour chaque paramètre obligatoire. Cependant, l'appel EC2 API de service Amazon échoue car le AWS service a toujours besoin de ces informations.

```
PS > Get-EC2InstanceAttribute -InstanceId $null -Attribute $null
WARNING: You are passing $null as a value for parameter Attribute which is marked as
 required.
In case you believe this parameter was incorrectly marked as required, report this by
 opening
an issue at https://github.com/aws/aws-tools-for-powershell/issues .
WARNING: You are passing $null as a value for parameter InstanceId which is marked as
 required.
In case you believe this parameter was incorrectly marked as required, report this by
 opening
an issue at https://github.com/aws/aws-tools-for-powershell/issues .

Get-EC2InstanceAttribute : The request must contain the parameter instanceId
```

## Suppression des fonctions obsolètes dans les versions antérieures

Les fonctionnalités suivantes étaient obsolètes dans les versions précédentes de AWS Tools for PowerShell et ont été supprimées dans la version 4 :

- Suppression du paramètre `-Terminate` de l'applet de commande `Stop-EC2Instance`. Utilisez `Remove-EC2Instance` à la place.
- Le `-ProfileName` paramètre a été supprimé de l'AWSCredentialapplet de commande Clear-. Utilisez `Remove-AWSCredentialProfile` à la place.
- Suppression des applets de commande `Import-EC2Instance` et `Import-EC2Volume`.

# Premiers pas avec AWS Tools for Windows PowerShell

Certaines rubriques de cette section décrivent les principes de base de l'utilisation des Outils pour Windows PowerShell une fois que vous avez installé les outils. Par exemple, elles expliquent comment indiquer les informations d'identification et la région AWS que les Outils pour Windows PowerShell devront utiliser lors de leur interaction avec AWS.

D'autres rubriques de cette section fournissent des informations sur les méthodes avancées de configuration des outils, de votre environnement et de vos projets.

Rubriques



## Configurez l'authentification des outils avec AWS

Vous devez définir la manière dont votre code s'authentifie AWS lorsque vous développez avec Services AWS. Vous pouvez configurer l'accès programmatique aux AWS ressources de différentes manières, en fonction de l'environnement et de l' AWS accès dont vous disposez.

Pour connaître les différentes méthodes d'authentification pour les outils PowerShell, consultez la section Authentification et accès dans le Guide de référence AWS des SDK et des outils.

Cette rubrique part du principe qu'un nouvel utilisateur se développe localement, qu'il n'a pas reçu de méthode d'authentification de la part de son employeur et qu'il l'utilisera AWS IAM Identity Center pour obtenir des informations d'identification temporaires. Si votre environnement s'écarte de ces hypothèses, certaines des informations contenues dans cette rubrique ne s'appliquent peut-être pas à votre cas ou vous ont déjà été communiquées.

La configuration de cet environnement nécessite plusieurs étapes, qui sont résumées comme suit :

1. Activation et configuration d’IAM Identity Center
2. Configurez les outils PowerShell pour utiliser IAM Identity Center.
3. Démarrer une session sur le portail AWS d'accès

## Activation et configuration d’IAM Identity Center

Pour être utilisé AWS IAM Identity Center, il doit d'abord être activé et configuré. Pour en savoir plus sur la procédure à suivre PowerShell, consultez l'étape 1 de la rubrique consacrée à l'authentification IAM Identity Center dans le Guide de référence AWS des SDK et des outils. Suivez particulièrement toutes les instructions nécessaires dans I do not have established access through IAM Identity Center.

## Configurez les outils PowerShell pour utiliser IAM Identity Center.

> **Note**
>
> À partir de la version 4.1.538 des outils pour PowerShell, la méthode recommandée pour configurer les informations d'identification SSO et démarrer une session de portail d' AWS accès consiste à utiliser les `Invoke-AWSSSOLogin`applets de commande `Initialize-AWSSSOConfiguration`et, comme décrit dans cette rubrique. Si vous n'avez pas accès à cette version des outils pour PowerShell (ou version ultérieure) ou si vous ne pouvez pas utiliser ces applets de commande, vous pouvez toujours effectuer ces tâches à l'aide du. AWS CLI Pour savoir comment procéder, voirUtilisez le AWS CLI pour vous connecter au portail.

La procédure suivante met à jour le AWS `config` fichier partagé avec les informations SSO que les outils PowerShell utilisent pour obtenir des informations d'identification temporaires. À la suite de cette procédure, une session du portail d' AWS accès est également démarrée. Si le `config` fichier partagé contient déjà des informations SSO et que vous souhaitez simplement savoir comment démarrer une session de portail d'accès à l'aide des outils pour PowerShell, consultez la section suivante de cette rubrique,Démarrer une session sur le portail AWS d'accès.

1. Si ce n'est pas déjà fait, ouvrez PowerShell et installez le AWS Tools for PowerShell en fonction de votre système d'exploitation et de votre environnement, y compris les applets de commande courants. Pour plus d'informations sur la procédure à utiliser, consultez Installation du AWS Tools for PowerShell.

Par exemple, si vous installez la version modulaire des outils pour PowerShell Windows, vous exécuterez probablement des commandes similaires aux suivantes :

```
Install-Module -Name AWS.Tools.Installer
Install-AWSToolsModule AWS.Tools.Common
```

2. Exécutez la commande suivante. Remplacez les valeurs de propriété d'exemple par des valeurs issues de votre configuration IAM Identity Center. Pour plus d'informations sur ces propriétés et sur la manière de les trouver, consultez les paramètres du fournisseur d'informations d'identification IAM Identity Center dans le Guide de référence AWS des SDK et des outils.

```
$params = @{
  ProfileName = 'my-sso-profile'
  AccountId = '111122223333'
  RoleName = 'SamplePermissionSet'
  SessionName = 'my-sso-session'
  StartUrl = 'https://provided-domain.awsapps.com/start'
  SSORegion = 'us-west-2'
  RegistrationScopes = 'sso:account:access'
};
Initialize-AWSSSOConfiguration @params
```

Vous pouvez également simplement utiliser l'applet de commande seule`Initialize-AWSSSOConfiguration`, et les outils vous demandent de PowerShell saisir les valeurs des propriétés.

Considérations relatives à la valeur de certaines propriétés :

- Si vous avez simplement suivi les instructions pour activer et configurer IAM Identity Center, la valeur de `-RoleName` pourrait être`PowerUserAccess`. Mais si vous avez créé un ensemble d'autorisations IAM Identity Center spécifiquement pour le PowerShell travail, utilisez-le plutôt.
- Assurez-vous d'utiliser l' Région AWS endroit où vous avez configuré IAM Identity Center.

3. À ce stade, le AWS `config` fichier partagé contient un profil appelé `my-sso-profile` avec un ensemble de valeurs de configuration qui peuvent être référencées à partir des outils pour PowerShell. Pour connaître l'emplacement de ce fichier, consultez Location of the shared files dans le manuel AWS SDKs and Tools Reference Guide.

The Tools for PowerShell utilise le fournisseur de jetons SSO du profil pour acquérir des informations d'identification avant d'envoyer des demandes à AWS. La `sso_role_name` valeur,

qui est un rôle IAM connecté à un ensemble d'autorisations IAM Identity Center, doit autoriser l'accès à l'utilisateur dans Services AWS votre application.

L'exemple suivant montre le profil créé à l'aide de la commande ci-dessus. Certaines valeurs des propriétés et leur ordre peuvent être différents dans votre profil actuel. La `sso-session` propriété du profil fait référence à la section nomméemy`-sso-session`, qui contient les paramètres permettant de lancer une session sur le portail d' AWS accès.

```
[profile my-sso-profile]
sso_account_id=111122223333
sso_role_name=SamplePermissionSet
sso_session=my-sso-session

[sso-session my-sso-session]
sso_region=us-west-2
sso_registration_scopes=sso:account:access
sso_start_url=https://provided-domain.awsapps.com/start/
```

4. Si vous avez déjà une session active sur le portail d' AWS accès, les PowerShell outils pour vous indiquent que vous êtes déjà connecté.

   Si ce n'est pas le cas, les outils pour PowerShell tenter d'ouvrir automatiquement la page d'autorisation SSO dans votre navigateur Web par défaut. Suivez les instructions de votre navigateur, qui peuvent inclure un code d'autorisation SSO, un nom d'utilisateur et un mot de passe, ainsi que l'autorisation d'accéder aux AWS IAM Identity Center comptes et aux ensembles d'autorisations.

   Les outils pour vous PowerShell indiquent que la connexion SSO a réussi.

## Démarrer une session sur le portail AWS d'accès

Avant d'exécuter des commandes d'accès Services AWS, vous avez besoin d'une session de portail d' AWS accès active afin que les outils PowerShell puissent utiliser l'authentification IAM Identity Center pour résoudre les informations d'identification. Pour vous connecter au portail AWS d'accès, exécutez la commande suivante PowerShell, où se `-ProfileName my-sso-profile` trouve le nom du profil créé dans le `config` fichier partagé lorsque vous avez suivi la procédure décrite dans la section précédente de cette rubrique.

```
Invoke-AWSSSOLogin -ProfileName my-sso-profile
```

Si vous avez déjà une session active sur le portail d' AWS accès, les PowerShell outils pour vous indiquent que vous êtes déjà connecté.

Si ce n'est pas le cas, les outils pour PowerShell tenter d'ouvrir automatiquement la page d'autorisation SSO dans votre navigateur Web par défaut. Suivez les instructions de votre navigateur, qui peuvent inclure un code d'autorisation SSO, un nom d'utilisateur et un mot de passe, ainsi que l'autorisation d'accéder aux AWS IAM Identity Center comptes et aux ensembles d'autorisations.

Les outils pour vous PowerShell indiquent que la connexion SSO a réussi.

Pour vérifier si vous avez déjà une session active, exécutez la commande suivante après avoir installé ou importé le `AWS.Tools.SecurityToken` module selon vos besoins.

```
Get-STSCallerIdentity -ProfileName my-sso-profile
```

La réponse à l'`Get-STSCallerIdentity`applet de commande indique le compte IAM Identity Center et l'ensemble d'autorisations configurés dans le fichier partagé. `config`

## Exemple

Voici un exemple d'utilisation d'IAM Identity Center avec les outils pour PowerShell. Cet exemple suppose que :

- Vous avez activé IAM Identity Center et l'avez configuré comme décrit précédemment dans cette rubrique. Les propriétés SSO se trouvent dans le `my-sso-profile` profil, qui a été configuré plus haut dans cette rubrique.
- Lorsque vous vous connectez via les `Invoke-AWSSSOLogin` applets de commande `Initialize-AWSSSOConfiguration` ou, l'utilisateur dispose au moins d'autorisations en lecture seule pour Amazon S3.
- Cet utilisateur peut consulter certains compartiments S3.

Installez ou importez le `AWS.Tools.S3` module selon vos besoins, puis utilisez la PowerShell commande suivante pour afficher la liste des compartiments S3.

```
Get-S3Bucket -ProfileName my-sso-profile
```

## Informations supplémentaires

- Pour plus d'options d'authentification pour les outils PowerShell, telles que l'utilisation de profils et de variables d'environnement, consultez le chapitre sur la configuration du Guide de référence AWS des SDK et des outils.
- Certaines commandes nécessitent AWS la spécification d'une région. Il existe plusieurs méthodes pour ce faire, notamment l'option d'`-Region`applet de commande, le `[default]` profil et la variable d'`AWS_REGION`environnement. Pour plus d'informations, consultez Spécifier AWS les régions ce guide et AWS Region dans le Guide de référence AWS des SDK et des outils.
- Pour en savoir plus sur les bonnes pratiques, consultez Bonnes pratiques de sécurité dans IAM dans le Guide de l'utilisateur IAM.
- Pour créer des AWS informations d'identification à court terme, consultez la section Informations d'identification de sécurité temporaires dans le guide de l'utilisateur IAM.
- Pour en savoir plus sur les autres fournisseurs d'informations d'identification, consultez Standardized credential providers dans le manuel AWS SDKs and Tools Reference Guide.

Rubriques

- Utilisez le AWS CLI pour vous connecter au portail

## Utilisez le AWS CLI pour vous connecter au portail

À partir de la version 4.1.538 des outils pour PowerShell, la méthode recommandée pour configurer les informations d'identification SSO et démarrer une session de portail d' AWS accès consiste à utiliser les `Invoke-AWSSSOLogin`applets de commande `Initialize-AWSSSOConfiguration`et, comme décrit dans. Configurez l'authentification des outils avec AWS Si vous n'avez pas accès à cette version des outils pour PowerShell (ou version ultérieure) ou si vous ne pouvez pas utiliser ces applets de commande, vous pouvez toujours effectuer ces tâches à l'aide du. AWS CLI

### Configurez les outils PowerShell pour utiliser IAM Identity Center via le AWS CLI.

Si ce n'est pas déjà fait, assurez-vous d'activer et de configurer IAM Identity Center avant de continuer.

Vous trouverez des informations sur la configuration des outils PowerShell pour utiliser IAM Identity Center via l' AWS CLI étape 2 de la rubrique relative à l'authentification IAM Identity Center dans le

Guide de référence AWS des SDK et des outils. Une fois cette configuration terminée, votre système doit contenir les éléments suivants :

- Le AWS CLI, que vous utilisez pour démarrer une session de portail d' AWS accès avant d'exécuter votre application.
- Le AWS `config` fichier partagé qui contient un `[default]`profil avec un ensemble de valeurs de configuration pouvant être référencées à partir des outils pour PowerShell. Pour connaître l'emplacement de ce fichier, consultez Location of the shared files dans le manuel AWS SDKs and Tools Reference Guide. The Tools for PowerShell utilise le fournisseur de jetons SSO du profil pour acquérir des informations d'identification avant d'envoyer des demandes à AWS. La `sso_role_name` valeur, qui est un rôle IAM connecté à un ensemble d'autorisations IAM Identity Center, doit autoriser l'accès à l'utilisateur dans Services AWS votre application.

  Le `config` fichier d'exemple suivant montre un `[default]` profil configuré avec un fournisseur de jetons SSO. Le paramètre `sso_session` du profil fait référence à la section `sso-session` nommée. La `sso-session` section contient les paramètres permettant de lancer une session sur le portail AWS d'accès.

```
[default]
sso_session = my-sso
sso_account_id = 111122223333
sso_role_name = SampleRole
region = us-east-1
output = json

[sso-session my-sso]
sso_region = us-east-1
sso_start_url = https://provided-domain.awsapps.com/start
sso_registration_scopes = sso:account:access
```

**Important**

Les modules suivants doivent être installés et importés dans votre PowerShell session pour que la résolution SSO puisse fonctionner :

- `AWS.Tools.SSO`
- `AWS.Tools.SSOOIDC`

Si vous utilisez une ancienne version des outils pour PowerShell et que vous ne disposez pas de ces modules, vous recevrez une erreur similaire à la suivante : « L'assemblage AWSSDK .SSOIDC est introuvable... ».

## Démarrer une session sur le portail AWS d'accès

Avant d'exécuter des commandes d'accès Services AWS, vous devez disposer d'une session de portail d' AWS accès active afin que les Outils pour Windows PowerShell puissent utiliser l'authentification IAM Identity Center pour résoudre les informations d'identification. En fonction de la durée de session que vous avez configurée, votre accès finira par expirer et les Outils pour Windows PowerShell rencontreront une erreur d'authentification. Pour vous connecter au portail AWS d'accès, exécutez la commande suivante dans le AWS CLI.

```
aws sso login
```

Puisque vous utilisez le `[default]` profil, il n'est pas nécessaire d'appeler la commande avec l'`--profile`option. Si la configuration de votre fournisseur de jetons SSO utilise un profil nommé, la commande l'est à la `aws sso login --profile` *`named-profile`* place. Pour plus d'informations sur les profils nommés, consultez la section Profils du Guide de référence AWS des SDK et des outils.

Pour vérifier si vous avez déjà une session active, exécutez la AWS CLI commande suivante (avec la même considération pour le profil nommé) :

```
aws sts get-caller-identity
```

La réponse à cette commande doit indiquer le compte IAM Identity Center et l'ensemble d'autorisations configurés dans le fichier partagé `config`.

> **Note**
>
> Si vous disposez déjà d'une session active sur le portail AWS d'accès et que vous l'exécutez`aws sso login`, il ne vous sera pas demandé de fournir des informations d'identification.

Le processus de connexion peut vous demander d'autoriser l' AWS CLI accès à vos données. Comme AWS CLI il repose sur le SDK pour Python, les messages d'autorisation peuvent contenir des variantes du `botocore` nom.

## Exemple

Voici un exemple d'utilisation d'IAM Identity Center avec les outils pour PowerShell. Cet exemple suppose que :

- Vous avez activé IAM Identity Center et l'avez configuré comme décrit précédemment dans cette rubrique. Les propriétés SSO se trouvent dans le profil `[default]`.
- Lorsque vous vous connectez via le en AWS CLI utilisant`aws sso login`, cet utilisateur dispose au moins d'autorisations en lecture seule pour Amazon S3.
- Cet utilisateur peut consulter certains compartiments S3.

Utilisez les PowerShell commandes suivantes pour afficher la liste des compartiments S3 :

```
Install-Module AWS.Tools.Installer
Install-AWSToolsModule S3
# And if using an older version of the AWS Tools for PowerShell:
Install-AWSToolsModule SSO, SSOOIDC

# In older versions of the AWS Tools for PowerShell, we're not invoking a cmdlet from
 these modules directly,
# so we must import them explicitly:
Import-Module AWS.Tools.SSO
Import-Module AWS.Tools.SSOOIDC

# Older versions of the AWS Tools for PowerShell don't support the SSO login flow, so
 login with the CLI
aws sso login

# Now we can invoke cmdlets using the SSO profile
Get-S3Bucket
```

Comme indiqué ci-dessus, étant donné que vous utilisez le `[default]` profil, il n'est pas nécessaire d'appeler l'`Get-S3Bucket`applet de commande avec l'`-ProfileName`option. Si la configuration de votre fournisseur de jetons SSO utilise un profil nommé, la commande est `Get-S3Bucket -`

`ProfileName` *`named-profile`*. Pour plus d'informations sur les profils nommés, consultez la section Profils du Guide de référence AWS des SDK et des outils.

### Informations supplémentaires

- Pour plus d'options d'authentification pour les outils PowerShell, telles que l'utilisation de profils et de variables d'environnement, consultez le chapitre de configuration du Guide de référence AWS des SDK et des outils.
- Certaines commandes nécessitent AWS la spécification d'une région. Il existe plusieurs méthodes pour ce faire, notamment l'option d'`-Region`applet de commande, le `[default]` profil et la variable d'`AWS_REGION`environnement. Pour plus d'informations, consultez Spécifier AWS les régions ce guide et AWS Region dans le Guide de référence AWS des SDK et des outils.
- Pour en savoir plus sur les bonnes pratiques, consultez Bonnes pratiques de sécurité dans IAM dans le Guide de l'utilisateur IAM.
- Pour créer des AWS informations d'identification à court terme, consultez la section Informations d'identification de sécurité temporaires dans le guide de l'utilisateur IAM.
- Pour en savoir plus sur les autres fournisseurs d'informations d'identification, consultez Standardized credential providers dans le manuel AWS SDKs and Tools Reference Guide.

## Spécifier AWS les régions

Il existe deux manières de spécifier la AWS région à utiliser lors de l'exécution de AWS Tools for PowerShell commandes :

- Utilisez le paramètre commun `-Region` pour les commandes individuelles.
- Utilisez la commande `Set-DefaultAWSRegion` pour définir une région par défaut pour toutes les commandes.

De nombreuses AWS applets de commande échouent si les Outils pour Windows ne PowerShell parviennent pas à déterminer la région à utiliser. Les exceptions incluent les applets de commande pour Amazon S3, Amazon SES et Amazon AWS Identity and Access Management, qui utilisent automatiquement par défaut un point de terminaison global.

Pour spécifier la région pour une seule AWS commande

Ajoutez le paramètre `-Region` à votre commande, par exemple :

```
PS > Get-EC2Image -Region us-west-2
```

Pour définir une région par défaut pour toutes les commandes AWS CLI de la session en cours

À partir de l'invite de PowerShell commande, tapez la commande suivante.

```
PS > Set-DefaultAWSRegion -Region us-west-2
```

> **Note**
>
> Cette valeur n'est valable que pour la session en cours. Pour appliquer le paramètre à toutes vos PowerShell sessions, ajoutez cette commande à votre PowerShell profil comme vous l'avez fait pour la `Import-Module` commande.

Pour afficher la région par défaut actuelle pour toutes les commandes de la AWS CLI

À partir de l'invite de PowerShell commande, tapez la commande suivante.

```
PS > Get-DefaultAWSRegion

Region    Name             IsShellDefault
------    ----             --------------
us-west-2 US West (Oregon) True
```

Pour effacer la région par défaut actuelle pour toutes les commandes AWS CLI

À partir de l'invite de PowerShell commande, tapez la commande suivante.

```
PS > Clear-DefaultAWSRegion
```

Pour consulter la liste de toutes les AWS régions disponibles

À partir de l'invite de PowerShell commande, tapez la commande suivante. Notez que la troisième colonne identifie la région par défaut de votre session en cours.

```
PS > Get-AWSRegion

Region         Name                      IsShellDefault
```

```
------          ----                        --------------
ap-east-1       Asia Pacific (Hong Kong)     False
ap-northeast-1  Asia Pacific (Tokyo)        False
...
us-east-2       US East (Ohio)              False
us-west-1       US West (N. California)     False
us-west-2       US West (Oregon)            True
...
```

> **Note**
>
> Certaines régions peuvent être prises en charge, mais elles ne sont pas incluses dans les résultats de l'applet de commande `Get-AWSRegion`. Par exemple, c'est parfois le cas des régions qui ne sont pas encore à l'échelle mondiale. Si vous ne pouvez pas spécifier une région en ajoutant le paramètre `-Region`, essayez plutôt d'indiquer la région dans un point de terminaison personnalisé, comme indiqué dans la section suivante.

## Spécification d'un point de terminaison personnalisé ou non standard

Spécifiez un point de terminaison personnalisé sous forme d'URL en ajoutant le paramètre `-EndpointUrl` commun à votre PowerShell commande Outils pour Windows, dans le format d'exemple suivant.

```
PS > Some-AWS-PowerShellCmdlet -EndpointUrl "custom endpoint URL" -Other -Parameters
```

Voici un exemple de commande à l'aide de l'applet de commande `Get-EC2Instance`. Le point de terminaison personnalisé se trouve dans la région `us-west-2` ou US West (Oregon) dans cet exemple, mais vous pouvez utiliser n'importe quelle autre région AWS prise en charge, y compris les régions qui ne sont pas répertoriées par `Get-AWSRegion`.

```
PS > Get-EC2Instance -EndpointUrl "https://service-custom-url.us-west-2.amazonaws.com"
 -InstanceID "i-0555a30a2000000e1"
```

## Informations supplémentaires

Pour plus d'informations sur AWS les régions, consultez AWS la section Région dans le Guide de référence AWS des SDK et des outils.

# Configurez l'identité fédérée avec le AWS Tools for PowerShell

Pour permettre aux utilisateurs de votre organisation d'accéder aux AWS ressources, vous devez configurer une méthode d'authentification standard et reproductible à des fins de sécurité, d'auditabilité, de conformité et pour permettre la séparation des rôles et des comptes. Bien qu'il soit courant de fournir aux utilisateurs la possibilité d'accéder AWS APIs, sans API accès fédéré, vous devez également créer AWS Identity and Access Management (IAM) utilisateurs, ce qui va à l'encontre de l'objectif de la fédération. Cette rubrique décrit la prise en charge du langage de balisage d'assertions de sécurité SAML (Security Assertion Markup Language) AWS Tools for PowerShell qui facilite votre solution d'accès fédéré.

SAMLle support vous AWS Tools for PowerShell permet de fournir à vos utilisateurs un accès fédéré aux AWS services. SAMLest un format XML basé sur un standard ouvert pour la transmission des données d'authentification et d'autorisation des utilisateurs entre les services, en particulier entre un fournisseur d'identité (tel qu'Active Directory Federation Services) et un fournisseur de services (tel que AWS). Pour plus d'informations à ce sujet SAML et SAMLsur son fonctionnement, consultez Wikipedia ou les spécifications SAML techniques sur le site Web de l'Organisation pour l'avancement des normes d'information structurées (OASIS). SAMLle support dans le AWS Tools for PowerShell est compatible avec SAML 2.0.

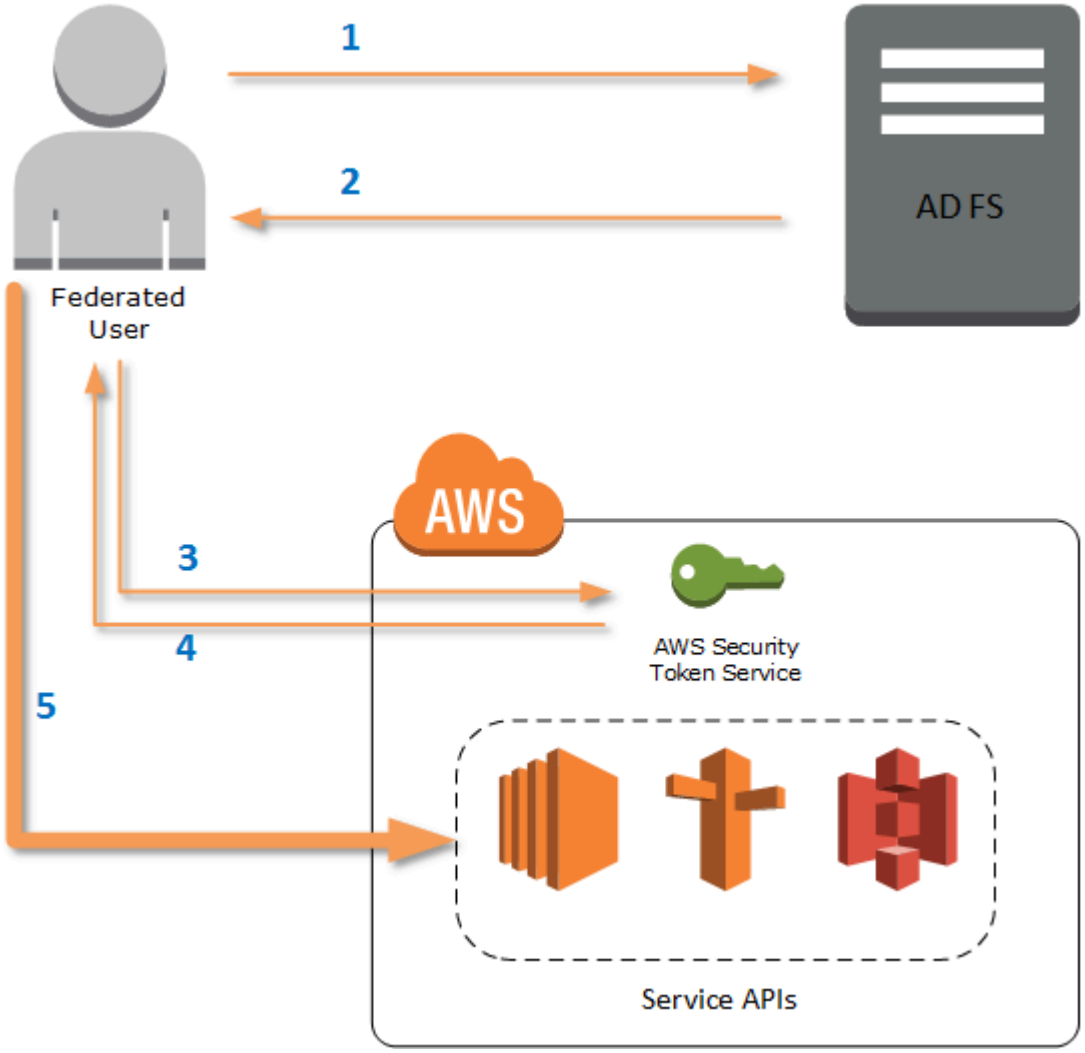
## Prérequis

Vous devez disposer des éléments suivants avant d'essayer d'utiliser le SAML support pour la première fois.

- Une solution de fédération d'identités correctement intégrée à votre compte AWS pour l'accès à la console en utilisant uniquement les informations d'identification de votre entreprise. Pour plus d'informations sur la manière de procéder spécifiquement pour les services de fédération Active Directory, consultez À propos de la fédération SAML 2.0 dans le guide de IAM l'utilisateur et le billet de blog intitulé Enabling Federation to AWS Using Windows Active Directory, AD FS et SAML 2.0. Même si l'article du blog couvre AD FS 2.0, les étapes sont similaires si vous exécutez AD FS 3.0.
- Version 3.1.31.0 ou ultérieure AWS Tools for PowerShell installée sur votre poste de travail local.

# Comment un utilisateur fédéré par identité obtient un accès fédéré au service AWS APIs

Le processus suivant décrit, de manière générale, comment un utilisateur Active Directory (AD) est fédéré par AD FS pour accéder aux AWS ressources.

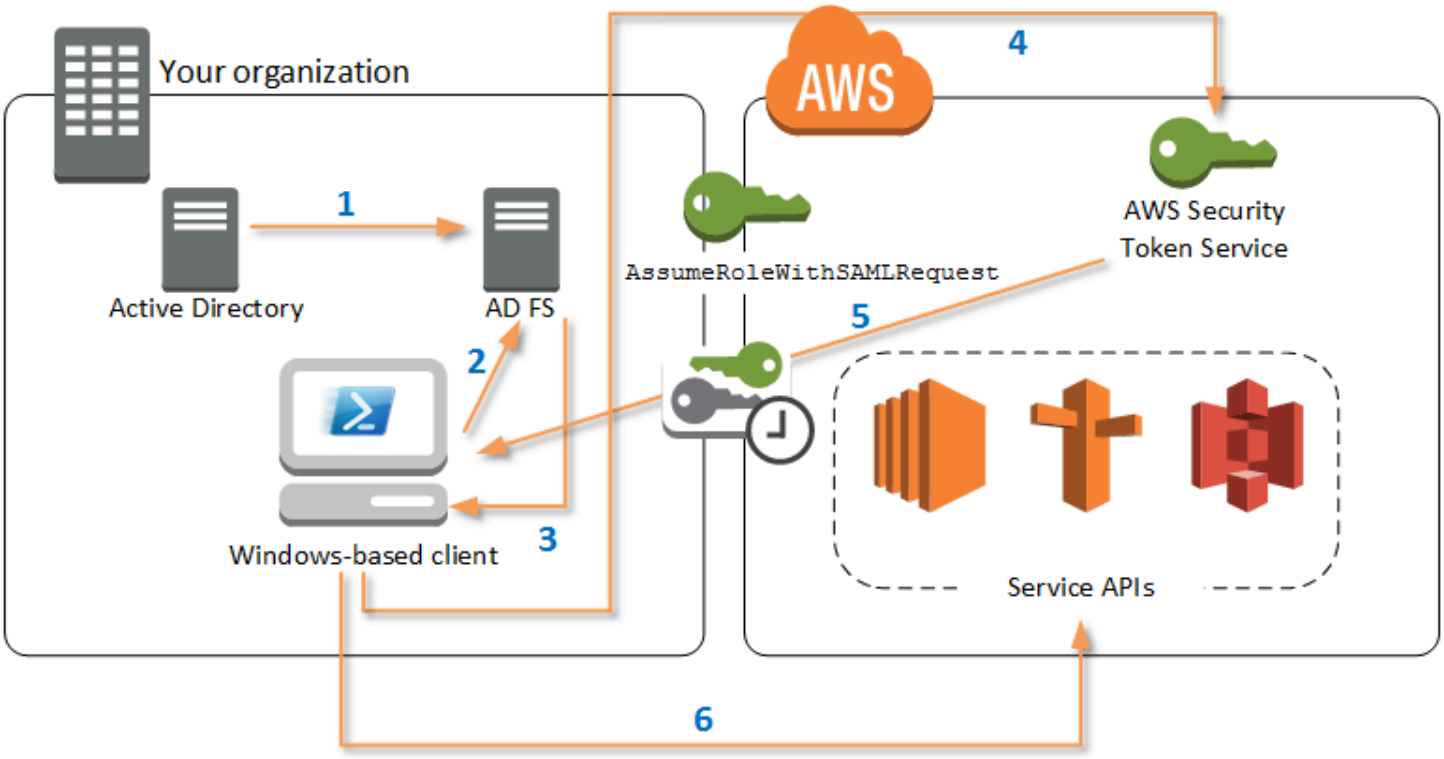



1. Le client sur l'ordinateur de l'utilisateur fédéré s'authentifie par rapport aux services AD FS.
2. Si l'authentification réussit, AD FS envoie une SAML assertion à l'utilisateur.
3. Le client de l'utilisateur envoie l'SAMLassertion à AWS Security Token Service (STS) dans le cadre d'une demande de SAML fédération.
4. STSrenvoie une SAML réponse contenant des informations d'identification AWS temporaires pour un rôle que l'utilisateur peut assumer.

5. L'utilisateur accède au AWS service APIs en incluant ces informations d'identification temporaires dans la demande faite par AWS Tools for PowerShell.

## Comment fonctionne SAML le Support dans le AWS Tools for PowerShell

Cette section décrit comment les AWS Tools for PowerShell applets de commande permettent de configurer la fédération d'identité SAML basée sur la base des utilisateurs.




1. AWS Tools for PowerShell s'authentifie auprès d'AD FS en utilisant les informations d'identification actuelles de l'utilisateur Windows, ou de manière interactive, lorsque l'utilisateur essaie d'exécuter une applet de commande nécessitant des informations d'identification pour être appelée. AWS
2. AD FS authentifie l'utilisateur.
3. AD FS génère une réponse d'authentification SAML 2.0 qui inclut une assertion ; le but de l'assertion est d'identifier et de fournir des informations sur l'utilisateur. AWS Tools for PowerShell extrait de l'SAMLassertion la liste des rôles autorisés de l'utilisateur.
4. AWS Tools for PowerShell transmet la SAML demande, y compris les Amazon Resource Names (ARN) du rôle demandé, à STS en effectuant l'`AssumeRoleWithSAMLRequest`APIappel.

5. Si la SAML demande est valide, STS renvoie une réponse contenant le AWS `AccessKeyIdSecretAccessKey`, et`SessionToken`. Ces informations d'identification durent 3 600 secondes (1 heure).
6. L'utilisateur dispose désormais d'informations d'identification valides lui permettant de travailler avec n'importe quel AWS service APIs auquel son rôle est autorisé à accéder. AWS Tools for PowerShell applique automatiquement ces informations d'identification pour tous les AWS API appels suivants et les renouvelle automatiquement lorsqu'elles expirent.

> **Note**
>
> Lorsque les informations d'identification arrivent à expiration, et que de nouvelles informations d'identification sont nécessaires, les AWS Tools for PowerShell sont automatiquement ré-authentifiés auprès d'AD FS et obtiennent de nouvelles informations d'identification pour une nouvelle heure. Pour les utilisateurs de comptes joints à un domaine, ce processus se produit silencieusement. Pour les comptes qui ne sont pas joints à un domaine, AWS Tools for PowerShell invite les utilisateurs à saisir leurs informations d'identification avant de pouvoir s'authentifier à nouveau.

## Comment utiliser les applets de PowerShell SAML commande de configuration

AWS Tools for PowerShell inclut deux nouvelles applets de commande qui fournissent une SAML assistance.

- `Set-AWSSamlEndpoint` configure votre point de terminaison AD FS, attribue un nom convivial au point de terminaison et, en option, décrit le type d'authentification du point de terminaison.
- `Set-AWSSamlRoleProfile` crée ou modifie un profil de compte utilisateur que vous souhaitez associer à un point de terminaison AD FS, identifié en spécifiant le nom convivial que vous avez fourni à l'applet de commande `Set-AWSSamlEndpoint`. Chaque profil de rôle correspond à un seul rôle qu'un utilisateur est autorisé à effectuer.

  Comme pour les profils AWS d'identification, vous attribuez un nom convivial au profil de rôle. Vous pouvez utiliser le même nom convivial pour l'`Set-AWSCredential`applet de commande ou comme valeur du `-ProfileName` paramètre pour toute applet de commande qui invoque le service. AWS APIs

Ouvrez une nouvelle AWS Tools for PowerShell session. Si vous utilisez la PowerShell version 3.0 ou une version plus récente, le AWS Tools for PowerShell module est automatiquement importé lorsque vous exécutez l'une de ses applets de commande. Si vous utilisez la PowerShell version 2.0, vous devez importer le module manuellement en exécutant l'applet de commande ``Import-Module``, comme illustré dans l'exemple suivant.

```
PS > Import-Module "C:\Program Files (x86)\AWS Tools\PowerShell\AWSPowerShell
\AWSPowerShell.psd1"
```

## Comment exécuter les applets de commande **Set-AWSSamlEndpoint** et **Set-AWSSamlRoleProfile**

1. Tout d'abord, configurez les paramètres de point de terminaison du système AD FS. Le moyen le plus simple de procéder consiste à stocker le point de terminaison dans une variable, comme indiqué dans cette étape. Assurez-vous de remplacer le compte fictif IDs et le nom d'hôte AD FS par vos propres compte IDs et nom d'hôte AD FS. Spécifiez le nom d'hôte AD FS dans le paramètre `Endpoint`.

   ```
   PS > $endpoint = "https://adfs.example.com/adfs/ls/IdpInitiatedSignOn.aspx?
   loginToRp=urn:amazon:webservices"
   ```

2. Pour créer les paramètres de point de terminaison, exécutez l'applet de commande `Set-AWSSamlEndpoint`, en spécifiant la valeur correcte du paramètre `AuthenticationType`. Les valeurs valides incluent `Basic`, `Digest`, `Kerberos`, `Negotiate` et `NTLM`. Si vous ne spécifiez pas ce paramètre, la valeur par défaut est `Kerberos`.

   ```
   PS > $epName = Set-AWSSamlEndpoint -Endpoint $endpoint -StoreAs ADFS-Demo -
   AuthenticationType NTLM
   ```

   L'applet de commande renvoie le nom amical que vous avez assigné en utilisant le paramètre `-StoreAs`. Ainsi, vous pouvez l'utiliser lorsque vous exécutez `Set-AWSSamlRoleProfile` à la ligne suivante.

3. Exécutez maintenant l'`Set-AWSSamlRoleProfile`applet de commande pour vous authentifier auprès du fournisseur d'identité AD FS et obtenir l'ensemble des rôles (dans l'SAMLassertion) que l'utilisateur est autorisé à exécuter.

   L'applet de commande `Set-AWSSamlRoleProfile` utilise l'ensemble de rôles retourné pour demander à l'utilisateur de sélectionner un rôle à associer au profil spécifié ou de valider que les

données de rôle fournies dans les paramètres sont présentes (si ce n'est pas le cas, l'utilisateur est invité à choisir). Si l'utilisateur est autorisé pour un seul rôle, l'applet de commande associe automatiquement le rôle au profil, sans intervention de l'utilisateur. Il n'est pas nécessaire de fournir les informations d'identification pour configurer un profil en vue d'une utilisation jointe à un domaine.

```
PS > Set-AWSSamlRoleProfile -StoreAs SAMLDemoProfile -EndpointName $epName
```

Pour les non-domain-joined comptes, vous pouvez également fournir des informations d'identification Active Directory, puis sélectionner un AWS rôle auquel l'utilisateur a accès, comme indiqué dans la ligne suivante. Cette solution est utile si vous avez différents comptes utilisateur Active Directory pour différencier les rôles au sein de votre entreprise (par exemple, les fonctions d'administration).

```
PS > $credential = Get-Credential -Message "Enter the domain credentials for the
 endpoint"
PS > Set-AWSSamlRoleProfile -EndpointName $epName -NetworkCredential $credential -
StoreAs SAMLDemoProfile
```

4. Dans tous les cas, l'applet de commande `Set-AWSSamlRoleProfile` vous invite à choisir le rôle qui doit être stocké dans le profil. L'exemple suivant montre deux rôles disponibles : `ADFS-Dev` et `ADFS-Production`. Les IAM rôles sont associés à vos identifiants de connexion AD par l'administrateur AD FS.

```
Select Role
Select the role to be assumed when this profile is active
[1] 1 - ADFS-Dev  [2] 2 - ADFS-Production  [?] Help (default is "1"):
```

Vous pouvez également spécifier un rôle sans invite, en entrant les paramètres `RoleARN`, `PrincipalARN` et `NetworkCredential` (ce dernier paramètre est facultatif). Si le rôle spécifié n'est pas répertorié dans l'assertion renvoyée par l'authentification, l'utilisateur est invité à choisir parmi les rôles disponibles.

```
PS > $params = @{ "NetworkCredential"=$credential,
 "PrincipalARN"="{arn:aws:iam::012345678912:saml-provider/ADFS}",
 "RoleARN"="{arn:aws:iam::012345678912:role/ADFS-Dev}"
}
PS > $epName | Set-AWSSamlRoleProfile @params -StoreAs SAMLDemoProfile1 -Verbose
```

5. Vous pouvez créer des profils pour tous les rôles en une seule commande en ajoutant le paramètre `StoreAllRoles`, comme indiqué dans le code suivant. Notez que le nom du rôle est utilisé comme nom de profil.

```
PS > Set-AWSSamlRoleProfile -EndpointName $epName -StoreAllRoles
ADFS-Dev
ADFS-Production
```

## Comment utiliser des profils de rôle pour exécuter des applets de commande nécessitant des informations d'identification AWS

Pour exécuter des applets de commande qui nécessitent des AWS informations d'identification, vous pouvez utiliser les profils de rôle définis dans le fichier d'informations d'identification AWS partagé. Fournissez le nom d'un profil de rôle `Set-AWSCredential` (ou comme valeur de n'importe quel `ProfileName` paramètre du AWS Tools for PowerShell) afin d'obtenir automatiquement des AWS informations d'identification temporaires pour le rôle décrit dans le profil.

Même si vous n'utilisez qu'un seul profil de rôle à la fois, vous pouvez basculer entre les profils au sein d'une session shell. L'applet de commande `Set-AWSCredential` n'authentifie pas et n'obtient pas les informations d'identification lorsque vous l'exécutez seule ; l'applet de commande enregistre que vous souhaitez utiliser un profil de rôle spécifié. Tant que vous n'exécutez pas une applet de commande nécessitant des informations d'identification AWS , aucune authentification ou demande d'informations d'identification ne se produit.

Vous pouvez désormais utiliser les AWS informations d'identification temporaires que vous avez obtenues avec le `SAMLDemoProfile` profil pour utiliser le AWS serviceAPIs. Les sections suivantes fournissent des exemples sur l'utilisation des profils de rôle.

### Exemple 1 : Définir un rôle par défaut avec **`Set-AWSCredential`**

Cet exemple définit un rôle par défaut pour une AWS Tools for PowerShell session en utilisant`Set-AWSCredential`. Ensuite, vous pouvez exécuter des applets de commande qui nécessitent des informations d'identification et sont autorisées par le rôle spécifié. Cet exemple répertorie toutes les instances Amazon Elastic Compute Cloud de la région USA Ouest (Oregon) associées au profil que vous avez spécifié avec l'applet de commande `Set-AWSCredential`.

```
PS > Set-AWSCredential -ProfileName SAMLDemoProfile
PS > Get-EC2Instance -Region us-west-2 | Format-Table -Property Instances,GroupNames
```

```
Instances                                                    GroupNames
---------                                                    ----------
{TestInstance1}                                              {default}
{TestInstance2}                                              {}
{TestInstance3}                                              {launch-wizard-6}
{TestInstance4}                                              {default}
{TestInstance5}                                              {}
{TestInstance6}                                              {AWS-OpsWorks-Default-
Server}
```

## Exemple 2 : Modifier les profils de rôle au cours d'une PowerShell session

Cet exemple répertorie tous les compartiments Amazon S3 disponibles dans le AWS compte du rôle associé au `SAMLDemoProfile` profil. L'exemple montre que même si vous avez peut-être utilisé un autre profil plus tôt dans votre AWS Tools for PowerShell session, vous pouvez modifier les profils en spécifiant une valeur différente pour le `-ProfileName` paramètre avec les applets de commande qui le prennent en charge. Il s'agit d'une tâche courante pour les administrateurs qui gèrent Amazon S3 depuis la ligne de PowerShell commande.

```
PS > Get-S3Bucket -ProfileName SAMLDemoProfile

CreationDate                                                 BucketName
------------                                                 ----------
7/25/2013 3:16:56 AM                                         amzn-s3-demo-bucket
4/15/2015 12:46:50 AM                                        amzn-s3-demo-bucket1
4/15/2015 6:15:53 AM                                         amzn-s3-demo-bucket2
1/12/2015 11:20:16 PM                                        amzn-s3-demo-bucket3
```

Notez que l'applet de commande `Get-S3Bucket` spécifie le nom du profil créé en exécutant l'applet de commande `Set-AWSSamlRoleProfile`. Cette commande peut être utile si vous avez défini un profil de rôle précédemment dans votre session (par exemple, en exécutant l'applet de commande `Set-AWSCredential`) et que vous voulez utiliser un autre rôle pour l'applet de commande `Get-S3Bucket`. Le gestionnaire de profils rend les informations d'identification temporaires accessibles à l'applet de commande `Get-S3Bucket`.

Bien que les informations d'identification expirent au bout d'une heure (limite imposée parSTS), les actualise AWS Tools for PowerShell automatiquement en demandant une nouvelle SAML assertion lorsque l'outil détecte que les informations d'identification actuelles ont expiré.

Pour les utilisateurs joints à un domaine, ce processus a lieu sans interruption, car l'identité Windows de l'utilisateur actif est utilisée lors de l'authentification. Pour les comptes non-domain-joined

utilisateur, AWS Tools for PowerShell affiche une demande PowerShell d'identification demandant le mot de passe utilisateur. L'utilisateur fournit les informations d'identification qui sont utilisées pour l'authentifier à nouveau et obtenir une nouvelle assertion.

### Exemple 3 : Obtenir les instances d'une région

L'exemple suivant répertorie toutes les EC2 instances Amazon de la région Asie-Pacifique (Sydney) associées au compte utilisé par le `ADFS-Production` profil. Il s'agit d'une commande utile pour renvoyer toutes les EC2 instances Amazon d'une région.

```
PS > (Get-Ec2Instance -ProfileName ADFS-Production -Region ap-southeast-2).Instances |
 Select InstanceType, @{Name="Servername";Expression={$_.tags | where key -eq "Name" |
 Select Value -Expand Value}}

 InstanceType                                                    Servername
 ------------                                                    ----------
 t2.small                                                        DC2
 t1.micro                                                        NAT1
 t1.micro                                                        RDGW1
 t1.micro                                                        RDGW2
 t1.micro                                                        NAT2
 t2.small                                                        DC1
 t2.micro                                                        BUILD
```

## Lectures complémentaires

Pour des informations générales sur la mise en œuvre de l'APIaccès fédéré, voir Comment implémenter une solution générale pour l'CLIaccès API fédéré/à l'aide SAML de la version 2.0.

Pour toute question ou commentaire d'assistance, rendez-vous sur les forums des AWS développeurs pour les PowerShell scripts ou. NETDéveloppement.

# Découverte d'applets de commande et alias

Cette section explique comment répertorier les services pris en charge par les AWS Tools for PowerShell, comment afficher l'ensemble d'applets de commande fourni par les AWS Tools for PowerShell en soutien de ces services, et comment trouver les autres noms d'applet de commande (également appelés alias) pour accéder à ces services.

# Découverte d'applets de commande

Toutes les opérations de service AWS (ou API) sont documentées dans le Guide de référence des API pour chaque service. Par exemple, consultez la Référence d'API IAM. Dans la plupart des cas, il existe une correspondance individuelle entre une API de service AWS et une applet de commande PowerShell AWS. Pour obtenir le nom de l'applet de commande qui correspond à un nom d'API de service AWS, exécutez l'applet de commande AWS `Get-AWSCmdletName` avec le paramètre `-ApiOperation` et le nom de l'API de service AWS. Par exemple, pour obtenir tous les noms d'applets de commande possibles correspondant aux API de service `DescribeInstances` AWS disponibles, exécutez la commande suivante :

```
PS > Get-AWSCmdletName -ApiOperation DescribeInstances

CmdletName        ServiceOperation    ServiceName                    CmdletNounPrefix
----------        ----------------    -----------                    ----------------
Get-EC2Instance   DescribeInstances   Amazon Elastic Compute Cloud   EC2
Get-GMLInstance   DescribeInstances   Amazon GameLift Service        GML
```

Le paramètre `-ApiOperation` est le paramètre par défaut. Vous pouvez donc omettre le nom du paramètre. L'exemple suivant est équivalent au précédent :

```
PS > Get-AWSCmdletName DescribeInstances
```

Si vous connaissez les noms de l'API et du service, vous pouvez inclure le paramètre `-Service`, ainsi que le préfixe du nom de l'applet de commande ou une partie du nom de service AWS. Par exemple, le préfixe du nom de l'applet de commande pour Amazon EC2 est `EC2`. Pour obtenir le nom de l'applet de commande qui correspond à l'API `DescribeInstances` dans le service Amazon EC2, exécutez l'une des commandes suivantes. Elles conduisent toutes à la même sortie :

```
PS > Get-AWSCmdletName -ApiOperation DescribeInstances -Service EC2
PS > Get-AWSCmdletName -ApiOperation DescribeInstances -Service Compute
PS > Get-AWSCmdletName -ApiOperation DescribeInstances -Service "Compute Cloud"

CmdletName        ServiceOperation    ServiceName                    CmdletNounPrefix
----------        ----------------    -----------                    ----------------
Get-EC2Instance   DescribeInstances   Amazon Elastic Compute Cloud   EC2
```

Les valeurs de paramètre de ces commandes sont sensibles à la casse.

Si vous ne connaissez pas le nom de l'API de service AWS souhaitée ou du service AWS, utilisez le paramètre `-ApiOperation`, ainsi que le modèle de correspondance et le paramètre `-MatchWithRegex`. Par exemple, pour obtenir tous les noms d'applets de commande disponibles qui contiennent `SecurityGroup`, exécutez la commande suivante :

```
PS > Get-AWSCmdletName -ApiOperation SecurityGroup -MatchWithRegex

CmdletName                                      ServiceOperation
   ServiceName                         CmdletNounPrefix
----------                                      ----------------
   -----------                         ----------------
Approve-ECCacheSecurityGroupIngress             AuthorizeCacheSecurityGroupIngress
   Amazon ElastiCache                  EC
Get-ECCacheSecurityGroup                        DescribeCacheSecurityGroups
   Amazon ElastiCache                  EC
New-ECCacheSecurityGroup                        CreateCacheSecurityGroup
   Amazon ElastiCache                  EC
Remove-ECCacheSecurityGroup                     DeleteCacheSecurityGroup
   Amazon ElastiCache                  EC
Revoke-ECCacheSecurityGroupIngress              RevokeCacheSecurityGroupIngress
   Amazon ElastiCache                  EC
Add-EC2SecurityGroupToClientVpnTargetNetwrk
 ApplySecurityGroupsToClientVpnTargetNetwork Amazon Elastic Compute Cloud       EC2
Get-EC2SecurityGroup                            DescribeSecurityGroups
   Amazon Elastic Compute Cloud       EC2
Get-EC2SecurityGroupReference                   DescribeSecurityGroupReferences
   Amazon Elastic Compute Cloud       EC2
Get-EC2StaleSecurityGroup                       DescribeStaleSecurityGroups
   Amazon Elastic Compute Cloud       EC2
Grant-EC2SecurityGroupEgress                    AuthorizeSecurityGroupEgress
   Amazon Elastic Compute Cloud       EC2
Grant-EC2SecurityGroupIngress                   AuthorizeSecurityGroupIngress
   Amazon Elastic Compute Cloud       EC2
New-EC2SecurityGroup                            CreateSecurityGroup
   Amazon Elastic Compute Cloud       EC2
Remove-EC2SecurityGroup                         DeleteSecurityGroup
   Amazon Elastic Compute Cloud       EC2
Revoke-EC2SecurityGroupEgress                   RevokeSecurityGroupEgress
   Amazon Elastic Compute Cloud       EC2
Revoke-EC2SecurityGroupIngress                  RevokeSecurityGroupIngress
   Amazon Elastic Compute Cloud       EC2
Update-EC2SecurityGroupRuleEgressDescription  UpdateSecurityGroupRuleDescriptionsEgress
   Amazon Elastic Compute Cloud       EC2
```

```
Update-EC2SecurityGroupRuleIngressDescription
 UpdateSecurityGroupRuleDescriptionsIngress  Amazon Elastic Compute Cloud       EC2
Edit-EFSMountTargetSecurityGroup              ModifyMountTargetSecurityGroups
   Amazon Elastic File System         EFS
Get-EFSMountTargetSecurityGroup               DescribeMountTargetSecurityGroups
   Amazon Elastic File System         EFS
Join-ELBSecurityGroupToLoadBalancer           ApplySecurityGroupsToLoadBalancer
   Elastic Load Balancing             ELB
Set-ELB2SecurityGroup                         SetSecurityGroups
   Elastic Load Balancing V2          ELB2
Enable-RDSDBSecurityGroupIngress              AuthorizeDBSecurityGroupIngress
   Amazon Relational Database Service RDS
Get-RDSDBSecurityGroup                        DescribeDBSecurityGroups
   Amazon Relational Database Service RDS
New-RDSDBSecurityGroup                        CreateDBSecurityGroup
   Amazon Relational Database Service RDS
Remove-RDSDBSecurityGroup                     DeleteDBSecurityGroup
   Amazon Relational Database Service RDS
Revoke-RDSDBSecurityGroupIngress              RevokeDBSecurityGroupIngress
   Amazon Relational Database Service RDS
Approve-RSClusterSecurityGroupIngress         AuthorizeClusterSecurityGroupIngress
   Amazon Redshift                    RS
Get-RSClusterSecurityGroup                    DescribeClusterSecurityGroups
   Amazon Redshift                    RS
New-RSClusterSecurityGroup                    CreateClusterSecurityGroup
   Amazon Redshift                    RS
Remove-RSClusterSecurityGroup                 DeleteClusterSecurityGroup
   Amazon Redshift                    RS
Revoke-RSClusterSecurityGroupIngress          RevokeClusterSecurityGroupIngress
   Amazon Redshift                    RS
```

Si vous connaissez le nom du service AWS, mais pas celui de l'API de service AWS, incluez le paramètre `-MatchWithRegex` et le paramètre `-Service` pour limiter la recherche à un seul service. Par exemple, pour obtenir tous les noms d'applets de commande qui contiennent `SecurityGroup` uniquement dans le service Amazon EC2, exécutez la commande suivante :

```
PS > Get-AWSCmdletName -ApiOperation SecurityGroup -MatchWithRegex -Service EC2

CmdletName                                    ServiceOperation
   ServiceName                 CmdletNounPrefix
----------                                    ----------------
   -----------                 ----------------
```

```
Add-EC2SecurityGroupToClientVpnTargetNetwrk
 ApplySecurityGroupsToClientVpnTargetNetwork Amazon Elastic Compute Cloud EC2
Get-EC2SecurityGroup                          DescribeSecurityGroups
   Amazon Elastic Compute Cloud EC2
Get-EC2SecurityGroupReference                 DescribeSecurityGroupReferences
   Amazon Elastic Compute Cloud EC2
Get-EC2StaleSecurityGroup                     DescribeStaleSecurityGroups
   Amazon Elastic Compute Cloud EC2
Grant-EC2SecurityGroupEgress                  AuthorizeSecurityGroupEgress
   Amazon Elastic Compute Cloud EC2
Grant-EC2SecurityGroupIngress                 AuthorizeSecurityGroupIngress
   Amazon Elastic Compute Cloud EC2
New-EC2SecurityGroup                          CreateSecurityGroup
   Amazon Elastic Compute Cloud EC2
Remove-EC2SecurityGroup                       DeleteSecurityGroup
   Amazon Elastic Compute Cloud EC2
Revoke-EC2SecurityGroupEgress                 RevokeSecurityGroupEgress
   Amazon Elastic Compute Cloud EC2
Revoke-EC2SecurityGroupIngress                RevokeSecurityGroupIngress
   Amazon Elastic Compute Cloud EC2
Update-EC2SecurityGroupRuleEgressDescription  UpdateSecurityGroupRuleDescriptionsEgress
   Amazon Elastic Compute Cloud EC2
Update-EC2SecurityGroupRuleIngressDescription
 UpdateSecurityGroupRuleDescriptionsIngress  Amazon Elastic Compute Cloud EC2
```

Si vous connaissez le nom de la commande AWS Command Line Interface (AWS CLI), vous pouvez utiliser le paramètre `-AwsCliCommand` et le nom de la commande AWS CLI souhaitée pour obtenir le nom de l'applet de commande correspondant à la même API. Par exemple, pour obtenir le nom de l'applet de commande qui correspond à l'appel de la commande `authorize-security-group-ingress` AWS CLI dans le service Amazon EC2, exécutez la commande suivante :

```
PS > Get-AWSCmdletName -AwsCliCommand "aws ec2 authorize-security-group-ingress"

CmdletName                    ServiceOperation              ServiceName
  CmdletNounPrefix
----------                    ----------------              -----------
  ----------------
Grant-EC2SecurityGroupIngress AuthorizeSecurityGroupIngress Amazon Elastic Compute
 Cloud EC2
```

L'applet de commande `Get-AWSCmdletName` a uniquement besoin du nom de la commande AWS CLI pour identifier le service et l'API AWS.

Pour obtenir la liste de toutes les applets de commande de Tools for PowerShell Core, exécutez l'applet de commande PowerShell `Get-Command`, comme illustré dans l'exemple suivant.

```
PS > Get-Command -Module AWSPowerShell.NetCore
```

Vous pouvez exécuter la même commande avec `-Module AWSPowerShell` pour afficher les applets de commande dans les AWS Tools for Windows PowerShell.

L'applet de commande `Get-Command` génère la liste des applets de commande dans l'ordre alphabétique. Notez que, par défaut, la liste est triée par verbe PowerShell, plutôt que par nom PowerShell.

Pour trier les résultats par service, exécutez la commande suivante :

```
PS > Get-Command -Module AWSPowerShell.NetCore | Sort-Object Noun,Verb
```

Pour filtrer les applets de commande renvoyées par l'applet de commande `Get-Command`, orientez la sortie vers l'applet de commande PowerShell `Select-String`. Par exemple, pour afficher l'ensemble des applets de commande qui fonctionnent avec les régions AWS, exécutez la commande suivante :

```
PS > Get-Command -Module AWSPowerShell.NetCore | Select-String region

Clear-DefaultAWSRegion
Copy-HSM2BackupToRegion
Get-AWSRegion
Get-DefaultAWSRegion
Get-EC2Region
Get-LSRegionList
Get-RDSSourceRegion
Set-DefaultAWSRegion
```

Vous pouvez également rechercher les applets de commande d'un service spécifique en filtrant le préfixe de service des noms d'applet de commande. Pour afficher la liste des préfixes de service disponibles, exécutez `Get-AWSPowerShellVersion -ListServiceVersionInfo`. L'exemple suivant renvoie les applets de commande qui prennent en charge le service Amazon CloudWatch Events.

```
PS > Get-Command -Module AWSPowerShell -Noun CWE*
```

```
CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Cmdlet          Add-CWEResourceTag                                 3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Disable-CWEEventSource                             3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Disable-CWERule                                    3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Enable-CWEEventSource                              3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Enable-CWERule                                     3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEEventBus                                    3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEEventBusList                                3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEEventSource                                 3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEEventSourceList                             3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEPartnerEventSource                          3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEPartnerEventSourceAccountList               3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEPartnerEventSourceList                      3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEResourceTag                                 3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWERule                                        3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWERuleDetail                                  3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWERuleNamesByTarget                           3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWETargetsByRule                               3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          New-CWEEventBus                                    3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          New-CWEPartnerEventSource                          3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Remove-CWEEventBus                                 3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Remove-CWEPartnerEventSource                       3.3.563.1
 AWSPowerShell.NetCore
```

```
Cmdlet          Remove-CWEPermission                               3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Remove-CWEResourceTag                              3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Remove-CWERule                                     3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Remove-CWETarget                                   3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Test-CWEEventPattern                               3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Write-CWEEvent                                     3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Write-CWEPartnerEvent                              3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Write-CWEPermission                                3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Write-CWERule                                      3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Write-CWETarget                                    3.3.563.1
 AWSPowerShell.NetCore
```

## Dénomination d'applets de commande et alias

Les applets de commande fournies par les AWS Tools for PowerShell pour chaque service sont basées sur les méthodes fournies par le kit SDK AWS du service. Cependant, en raison des conventions de dénomination obligatoires de PowerShell, le nom d'une applet de commande peut être différent du nom de l'appel d'API ou de la méthode correspondante. Par exemple, l'applet de commande `Get-EC2Instance` est basée sur la méthode Amazon EC2`DescribeInstances`.

Dans certains cas, le nom de l'applet de commande peut être similaire à un nom de méthode, mais il peut en fait exécuter une fonction différente. Par exemple, la méthode Amazon S3`GetObject` récupère un objet Amazon S3. Cependant, l'applet de commande `Get-S3Object` renvoie les informations sur un objet Amazon S3 plutôt que l'objet lui-même.

```
PS > Get-S3Object -BucketName text-content -Key aws-tech-docs

ETag         : "df000002a0fe0000f3c000004EXAMPLE"
BucketName   : aws-tech-docs
Key          : javascript/frameset.js
LastModified : 6/13/2011 1:24:18 PM
Owner        : Amazon.S3.Model.Owner
Size         : 512
```

```
StorageClass : STANDARD
```

Pour obtenir un objet S3 avec les AWS Tools for PowerShell, exécutez l'applet de commande `Read-S3Object` :

```
PS > Read-S3Object -BucketName text-content -Key text-object.txt -file c:\tmp\text-object-download.text

Mode          LastWriteTime            Length Name
----          -------------            ------ ----
-a---         11/5/2012   7:29 PM      20622  text-object-download.text
```

> **Note**
>
> L'aide de l'applet de commande d'une applet de commande AWS fournit le nom de l'API SDK AWS sur laquelle l'applet de commande est basée.
> Pour plus d'informations sur les verbes PowerShell standard et leurs significations, consultez Approved Verbs for PowerShell Commands.

Toutes les applets de commande AWS qui utilisent le verbe `Remove`, ainsi que l'applet de commande `Stop-EC2Instance` quand vous ajoutez le paramètre `-Terminate`, demandent une confirmation avant de continuer. Pour contourner la confirmation, ajoutez le paramètre `-Force` à votre commande.

> **Important**
>
> Les applets de commande AWS ne prennent pas en charge le commutateur `-WhatIf`.

## Alias

Le programme d'installation AWS Tools for PowerShell installe un fichier d'alias contenant les alias d'un grand nombre d'applets de commande AWS. Vous trouverez peut-être ces alias plus intuitifs que les noms d'applets de commande. Par exemple, les noms de services et les noms de méthode SDK AWS remplacent les noms et les verbes PowerShell dans certains alias. L'alias `EC2-DescribeInstances` en est un exemple.

D'autres alias utilisent des verbes qui, bien qu'ils ne suivent pas les conventions PowerShell standard, peuvent être plus descriptifs de l'opération réelle. Par exemple, le fichier d'alias associe l'alias `Get-S3Content` à l'applet de commande `Read-S3Object`.

```
PS > Set-Alias -Name Get-S3Content -Value Read-S3Object
```

Le fichier d'alias se trouve dans le répertoire d'installation des AWS Tools for PowerShell. Pour charger les alias dans votre environnement, effectuez un appel de source dot source du fichier. L'exemple suivant est un exemple basé sur Windows.

```
PS > . "C:\Program Files (x86)\AWS Tools\PowerShell\AWSPowershell\AWSAliases.ps1"
```

Pour un shell Linux ou macOS, il peut se présenter comme suit :

```
. ~/.local/share/powershell/Modules/AWSPowerShell.NetCore/3.3.563.1/AWSAliases.ps1
```

Pour afficher tous les alias des AWS Tools for PowerShell, exécutez la commande suivante. Cette commande utilise l'alias `?` de l'applet de commande PowerShell `Where-Object` et la propriété `Source` pour filtrer uniquement les alias provenant du module `AWSPowerShell.NetCore`.

```
PS > Get-Alias | ? Source -like "AWSPowerShell.NetCore"

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Alias           Add-ASInstances                                    3.3.343.0
 AWSPowerShell
Alias           Add-CTTag                                          3.3.343.0
 AWSPowerShell
Alias           Add-DPTags                                         3.3.343.0
 AWSPowerShell
Alias           Add-DSIpRoutes                                     3.3.343.0
 AWSPowerShell
Alias           Add-ELBTags                                        3.3.343.0
 AWSPowerShell
Alias           Add-EMRTag                                         3.3.343.0
 AWSPowerShell
Alias           Add-ESTag                                          3.3.343.0
 AWSPowerShell
Alias           Add-MLTag                                          3.3.343.0
 AWSPowerShell
```

```
Alias           Clear-AWSCredentials                            3.3.343.0
 AWSPowerShell
Alias           Clear-AWSDefaults                               3.3.343.0
 AWSPowerShell
Alias           Dismount-ASInstances                            3.3.343.0
 AWSPowerShell
Alias           Edit-EC2Hosts                                   3.3.343.0
 AWSPowerShell
Alias           Edit-RSClusterIamRoles                          3.3.343.0
 AWSPowerShell
Alias           Enable-ORGAllFeatures                           3.3.343.0
 AWSPowerShell
Alias           Find-CTEvents                                   3.3.343.0
 AWSPowerShell
Alias           Get-ASACases                                    3.3.343.0
 AWSPowerShell
Alias           Get-ASAccountLimits                             3.3.343.0
 AWSPowerShell
Alias           Get-ASACommunications                           3.3.343.0
 AWSPowerShell
Alias           Get-ASAServices                                 3.3.343.0
 AWSPowerShell
Alias           Get-ASASeverityLevels                           3.3.343.0
 AWSPowerShell
Alias           Get-ASATrustedAdvisorCheckRefreshStatuses       3.3.343.0
 AWSPowerShell
Alias           Get-ASATrustedAdvisorChecks                     3.3.343.0
 AWSPowerShell
Alias           Get-ASATrustedAdvisorCheckSummaries             3.3.343.0
 AWSPowerShell
Alias           Get-ASLifecycleHooks                            3.3.343.0
 AWSPowerShell
Alias           Get-ASLifecycleHookTypes                        3.3.343.0
 AWSPowerShell
Alias           Get-AWSCredentials                              3.3.343.0
 AWSPowerShell
Alias           Get-CDApplications                              3.3.343.0
 AWSPowerShell
Alias           Get-CDDeployments                               3.3.343.0
 AWSPowerShell
Alias           Get-CFCloudFrontOriginAccessIdentities          3.3.343.0
 AWSPowerShell
Alias           Get-CFDistributions                             3.3.343.0
 AWSPowerShell
```

```
Alias           Get-CFGConfigRules                              3.3.343.0
 AWSPowerShell
Alias           Get-CFGConfigurationRecorders                   3.3.343.0
 AWSPowerShell
Alias           Get-CFGDeliveryChannels                         3.3.343.0
 AWSPowerShell
Alias           Get-CFInvalidations                             3.3.343.0
 AWSPowerShell
Alias           Get-CFNAccountLimits                            3.3.343.0
 AWSPowerShell
Alias           Get-CFNStackEvents                              3.3.343.0
 AWSPowerShell

...
```

Pour ajouter vos propres alias à ce fichier, vous devrez peut-être augmenter la valeur de la`$MaximumAliasCount` variable de préférence PowerShell pour obtenir une valeur supérieure à 5 500. La valeur par défaut est 4 096 et vous pouvez l'augmenter jusqu'à 32 768 au maximum. Pour ce faire, exécutez la commande suivante.

```
PS > $MaximumAliasCount = 32768
```

Pour vérifier que votre modification a abouti, saisissez le nom de la variable afin d'afficher sa valeur actuelle.

```
PS > $MaximumAliasCount
32768
```

## Mise en pipeline et $AWSHistory

Pour les appels de service AWS qui renvoient des collections, les objets de la collection sont désormais toujours énumérés sur le pipeline. Les objets de résultat qui contiennent des champs supplémentaires au-delà de la collection et qui ne sont pas des champs de contrôle de pagination ont ces champs ajoutés comme propriétés Note pour les appels. Ces propriétés Note sont enregistrées dans la nouvelle variable de session `$AWSHistory`, si vous devez accéder à ces données. La variable `$AWSHistory` est décrite dans la section suivante.

> **Note**
>
> Dans les versions des Tools for Windows PowerShelltait antérieures à v1.1, l'objet collection lui-même é émis, ce qui nécessitait l'utilisation de foreach {$_.getenumerator()} pour poursuivre la mise en pipeline.

Exemples

L'exemple suivant renvoie une liste de régions AWS et d'AMI (Amazon EC2 machine images) dans chaque région.

```
PS > Get-AWSRegion | % { Echo $_.Name; Get-EC2Image -Owner self -Region $_ }
```

L'exemple suivant montre comment arrêter toutes les instances Amazon EC2 de la région par défaut actuelle.

```
PS > Get-EC2Instance | Stop-EC2Instance
```

Dans la mesure où les collections sont énumérées sur le pipeline, la sortie d'une applet de commande fournie peut être `$null`, un seul objet ou une collection. S'il s'agit d'une collection, vous pouvez utiliser la propriété `.Count` pour déterminer la taille de la collection. Cependant, la propriété `.Count` n'est pas présente lorsqu'un seul objet est émis. Si votre script a besoin de déterminer, de manière cohérente, le nombre d'objets émis, vous pouvez utiliser la propriété `EmittedObjectsCount` de la valeur de la dernière commande dans `$AWSHistory`.

## $AWSHistory

Pour une meilleure prise en charge du pipeline, la sortie des applets de commande AWS n'est plus remodelée pour inclure la réponse du service et les instances de résultat comme propriétés Note sur l'objet de collection émis. Au lieu de cela, pour ces appels qui émettent une seule collection comme sortie, la collection est désormais énumérée dans le pipeline PowerShell. Cela signifie que le SDK AWS et les données des résultats ne peuvent pas exister dans le pipeline, car il n'y a pas d'objet de collection contenant auquel ils peuvent être attachés.

Bien que la plupart des utilisateurs n'aient probablement pas besoin de ces données, elles peuvent être utiles à des fins de diagnostic, car vous pouvez voir exactement ce qui a été envoyé et reçu vers ou depuis les appels de service AWS sous-jacents effectués par l'applet de commande.

À partir de la version 1.1, ces données, entre autres, sont désormais disponibles dans une nouvelle variable shell nommée `$AWSHistory`. Cette variable conserve un enregistrement des appels d'applets de commande AWS et les réponses de services reçues pour chaque appel. Le cas échéant, cet historique peut également être configuré pour enregistrer aussi les demandes de service que chaque applet de commande a effectuées. Des données utiles supplémentaires, telles que la durée d'exécution globale de l'applet de commande, peuvent aussi être obtenus à partir de chaque entrée. Pour des raisons de sécurité, par défaut, les demandes et les réponses contenant des données sensibles ne sont pas enregistrées. Toutefois, l'historique peut être configuré pour remplacer ce comportement si nécessaire. Pour plus d'informations, consultez l'applet de commande `Set-AWSHistoryConfiguration` présentée ci-dessous.

Chaque entrée de la liste `$AWSHistory.Commands` est de type `AWSCmdletHistory`. Ce type comporte les membres utiles suivants :

CmdletName

Nom de l'applet de commande.

CmdletStart

Date et heure auxquelles l'applet de commande a été exécutée.

CmdletEnd

Date et heure auxquelles l'applet de commande a terminé l'ensemble du traitement.

Requêtes

Si l'enregistrement de la demande est activé, la liste des dernières demandes de service.

Réponses

Liste des dernières réponses de service reçues.

LastServiceResponse

Annotations pour renvoyer la dernière réponse du service.

LastServiceRequest

Annotations pour renvoyer la dernière demande du service, le cas échéant.

Notez que la variable `$AWSHistory` ne sera pas créée avant qu'une applet de commande AWS effectuant un appel de service soit utilisée. Elle est analysée comme ayant la valeur $null jusqu'à ce stade.

> **Note**
>
> Les versions antérieures des Tools for Windows PowerShell émettaient les données relatives aux réponses de service en tant que propriétés `Note` sur l'objet renvoyé. Ces données sont désormais disponibles dans les entrées de réponse enregistrées pour chaque appel de la liste.

## Définir AWSHistoryConfiguration

Un appel d'applet de commande peut contenir zéro ou plusieurs entrées de demande et réponse de service. Pour limiter l'impact sur la mémoire, la liste `$AWSHistory` conserve par défaut un enregistrement des seules cinq dernières exécutions d'applet de commande ; et pour chacune, les cinq dernières réponses de service (et, si cette option est activée, les cinq dernières demandes de service). Vous pouvez modifier ces limites par défaut en exécutant l'applet de commande `Set-AWSHistoryConfiguration`. Vous pouvez ainsi contrôler la taille de la liste et si les demandes de service sont également enregistrées :

```
PS > Set-AWSHistoryConfiguration -MaxCmdletHistory <value> -MaxServiceCallHistory
 <value> -RecordServiceRequests -IncludeSensitiveData
```

Tous les paramètres sont facultatifs.

Le paramètre `MaxCmdletHistory` définit le nombre maximal d'applets de commande qui peuvent être suivis à tout moment. La valeur 0 désactive l'enregistrement de l'activité des applets de commande AWS. Le paramètre `MaxServiceCallHistory` définit le nombre maximal de réponses de service (et/ou de demandes) qui sont suivies pour chaque applet de commande. Le paramètre `RecordServiceRequests`, s'il est spécifié, active le suivi des demandes de service pour chaque applet de commande. Le paramètre `IncludeSensitiveData`, s'il est spécifié, active le suivi des demandes et réponses de service (si elles sont suivies) qui contiennent des données sensibles pour chaque applet de commande.

En cas d'exécution sans paramètres, `Set-AWSHistoryConfiguration` désactive simplement tout enregistrement de demandes précédentes et laisse inchangées les tailles de liste actuelles.

Pour effacer toutes les entrées de la liste d'historique actuelle, exécutez l'applet de commande `Clear-AWSHistory`.

## **$AWSHistory**Exemples

Énumérez les détails des applets de commande AWS conservées dans la liste du pipeline.

```
PS > $AWSHistory.Commands
```

Accédez aux détails de la dernière applet de commande AWS qui a été exécutée :

```
PS > $AWSHistory.LastCommand
```

Accédez aux détails de la dernière réponse de service reçue par la dernière applet de commande AWS exécutée. Si une applet de commande AWS pagine la sortie, elle peut effectuer plusieurs appels de service pour obtenir toutes les données ou la quantité maximale de données (déterminée par les paramètres de l'applet de commande).

```
PS > $AWSHistory.LastServiceResponse
```

Accédez aux détails de la dernière demande effectuée (à nouveau, une applet de commande peut effectuer plus d'une demande si elle pagine au nom de l'utilisateur). Génère $null, à moins que le suivi de demande de service ne soit activé.

```
PS > $AWSHistory.LastServiceRequest
```

### Achèvement de page automatique pour les opérations qui retournent plusieurs pages

Pour les API de service qui imposent un nombre maximal de retours d'objet par défaut pour un appel donné ou qui prennent en charge les jeux de résultats paginables, toutes les applets de commande effectuent un achèvement de page par défaut. Chaque applet de commande effectue autant d'appels que nécessaire en votre nom pour renvoyer l'ensemble complet des données au pipeline.

Dans l'exemple suivant, qui utilise `Get-S3Object`, la variable `$c` contient les instances de `S3Object` de chaque clé du compartiment `test`, potentiellement un très large ensemble de données.

```
PS > $c = Get-S3Object -BucketName test
```

Si vous souhaitez conserver le contrôle de la quantité de données renvoyées, vous pouvez utiliser des paramètres sur les applets de commande individuelles (par exemple, `MaxKey` sur `Get-`

S3Object) ou traiter explicitement la pagination vous-même en utilisant une combinaison des paramètres de pagination sur les applets de commande, et les données placées dans la variable $AWSHistory pour obtenir les données du jeton suivant du service. L'exemple suivant utilise le paramètre MaxKeys pour limiter le nombre d'instances S3Object renvoyées aux seules premières 500 du compartiment.

```
PS > $c = Get-S3Object -BucketName test -MaxKey 500
```

Pour savoir si d'autres données sont disponibles, mais n'ont pas été retournées, utilisez l'entrée de la variable de session $AWSHistory qui a enregistré les appels de service effectués par l'applet de commande.

Si l'expression suivante a la valeur $true, vous pouvez trouver le marqueur next du prochain ensemble de résultats à l'aide de $AWSHistory.LastServiceResponse.NextMarker :

```
$AWSHistory.LastServiceResponse -ne $null &&
 $AWSHistory.LastServiceResponse.IsTruncated
```

Pour contrôler manuellement la pagination avec Get-S3Object, utilisez une combinaison des paramètres MaxKey et Marker de l'applet de commande et les annotations IsTruncated/NextMarker de la dernière réponse enregistrée. Dans l'exemple suivant, la variable $c contient jusqu'à un maximum de 500 instances S3Object pour les 500 objets suivants trouvés dans le compartiment après le début du marqueur de préfixe de clé spécifié.

```
PS > $c = Get-S3Object -BucketName test -MaxKey 500 -Marker
 $AWSHistory.LastServiceResponse.NextMarker
```

# Résolution des informations d'identification et des profils

## Ordre de recherche des informations d'identification

Lorsque vous exécutez une commande, les AWS Tools for PowerShell recherchent les informations d'identification dans l'ordre suivant. Ils s'arrêtent lorsqu'ils trouvent des informations d'identification utilisables.

1. Informations d'identification littérales intégrées en tant que paramètres dans la ligne de commande.

Nous vous recommandons vivement d'utiliser des profils plutôt que de placer des informations d'identification littérales dans vos lignes de commande.

2. Un nom de profil ou un emplacement de profil spécifié.
   - Si vous spécifiez uniquement un nom de profil, la commande recherchera le profil spécifié dans le magasin SDK AWS et, s'il n'existe pas, le profil spécifié dans le fichier d'informations d'identification partagées AWS de l'emplacement par défaut.
   - Si vous spécifiez uniquement un emplacement de profil, la commande recherchera le profil `default` dans ce fichier d'informations d'identification.
   - Si vous spécifiez à la fois un nom et un emplacement, la commande recherchera le profil spécifié dans ce fichier d'informations d'identification.

   Si le profil ou l'emplacement spécifié est introuvable, la commande lèvera une exception. La recherche exécute les étapes suivantes uniquement si vous n'avez pas spécifié de profil ou d'emplacement.
3. Informations d'identification spécifiées par le paramètre `-Credential`.
4. Le profil de session, s'il en existe un.
5. Le profil par défaut, dans l'ordre suivant :
   a. Le profil `default` dans le magasin SDK AWS.
   b. Le profil `default` dans le fichier d'informations d'identification partagées AWS.
   c. Le profil `AWS PS Default` dans le magasin SDK AWS.
6. Si la commande s'exécute sur une instance Amazon EC2 configurée pour utiliser un rôle IAM, les informations d'identification temporaires de l'instance EC2 auxquelles le système a accédé à partir du profil d'instance.

   Pour en savoir plus sur l'utilisation des rôles IAM pour les instances Amazon EC2, consultez la section AWS SDK for .NET.

Si cette recherche échoue à retrouver les informations d'identification spécifiées, la commande lèvera une exception.

## Informations supplémentaires sur les utilisateurs et les rôles

Pour exécuter les commandes des Outils pour PowerShell sur AWS, vous devez disposer à la fois des utilisateurs, des ensembles d'autorisations et des fonctions du service adaptés à vos tâches.

Les utilisateurs, les ensembles d'autorisations et les rôles de service spécifiques que vous créez, ainsi que la manière dont vous les utilisez, dépendront de vos besoins. Vous trouverez ci-dessous des informations complémentaires indiquant pourquoi ils peuvent être utilisés et comment les créer.

## Utilisateurs et ensembles d'autorisations

Bien qu'il soit possible d'accéder aux services AWS en utilisant un compte utilisateur IAM avec des informations d'identification à long terme, il ne s'agit plus d'une bonne pratique et est à éviter. Même pendant le développement, il est recommandé de créer des utilisateurs et des ensembles d'autorisations dans AWS IAM Identity Center et d'utiliser des informations d'identification temporaires fournies par une source d'identité.

Pour le développement, vous pouvez employer l'utilisateur que vous avez créé ou qui vous a été attribué dans Configuration de l'authentification des outils. Si vous disposez des autorisations appropriées dans la AWS Management Console, vous pouvez également créer différents ensembles d'autorisations de moindre privilège pour cet utilisateur ou créer des utilisateurs spécialement pour les projets de développement, en fournissant des ensembles d'autorisations de moindre privilège. Le plan d'action que vous choisissez éventuellement dépend de votre situation.

Pour plus d'informations sur ces utilisateurs et ensembles d'autorisations et sur la façon de les créer, consultez Authentication and access dans le manuel AWS SDKs and Tools Reference Guide et Getting started dans le manuel AWS IAM Identity Center User Guide.

## Rôles de service

Vous pouvez configurer un rôle de service AWS pour accéder aux services AWS pour le compte des utilisateurs. Ce type d'accès est approprié si plusieurs personnes doivent exécuter votre application à distance ; par exemple, sur une instance Amazon EC2 que vous avez créée à cette fin.

Le processus de création d'un rôle de service varie en fonction de la situation, mais il consiste principalement en ce qui suit.

1. Connectez-vous à l'outil AWS Management Console, puis ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Cliquez sur Rôles, puis sur Créer un rôle.
3. Choisissez Service AWS, recherchez et sélectionnez EC2 (par exemple), puis choisissez le cas d'utilisation EC2 (par exemple).
4. Choisissez Suivant et sélectionnez les politiques appropriées pour les services AWS que votre application utilisera.

> **Warning**
>
> Ne choisissez PAS la politique AdministratorAccess, car elle active des autorisations de lecture et d'écriture pour presque tout dans votre compte.

5. Choisissez Next (Suivant). Saisissez un Nom de rôle, une Description et les balises de votre choix.

   Vous trouverez des informations sur les balises dans Contrôle de l'accès aux ressources AWS à l'aide de balises dans le Guide de l'utilisateur IAM.

6. Sélectionnez Create role (Créer un rôle).

Vous trouverez des informations générales sur les rôles IAM dans la section Identités IAM (utilisateurs, groupes d'utilisateurs et rôles) du Guide de l'utilisateur IAM. Vous trouverez des informations détaillées sur les rôles dans la rubrique Rôles IAM.

## Utilisation d’informations d'identification existantes

Les rubriques de cette section fournissent des informations sur l'utilisation d'informations d'identification à long terme ou à court terme sans utiliser AWS IAM Identity Center.

> **Warning**
>
> Afin d’éviter les risques de sécurité, n'employez pas les utilisateurs IAM pour l'authentification lorsque vous développez des logiciels spécialisés ou lorsque vous travaillez avec des données réelles. Préférez la fédération avec un fournisseur d'identité tel que AWS IAM Identity Center.

> **Note**
>
> Les informations contenues dans ces rubriques s’appliquent à des situations où vous devez obtenir et gérer manuellement des informations d'identification à court ou à long terme. Pour plus d'informations sur les informations d'identification à court et à long terme, consultez Other ways to authenticate dans le manuel AWS SDKs and Tools Reference Guide.

Pour les bonnes pratiques en matière de sécurité, utilisez AWS IAM Identity Center comme décrit dans Configuration de l'authentification des outils.

## Avertissements et conseils importants concernant les informations d'identification

Avertissements concernant les informations d'identification

- N'utilisez PAS les informations d'identification root de votre compte pour accéder aux ressources AWS. Ces informations d'identification offrent un accès illimité au compte et sont difficiles à révoquer.
- N'insérez PAS de clés d'accès littérales ou d'informations d'identification dans vos commandes ou scripts. Si vous le faites, vous risquez d'exposer accidentellement vos informations d'identification.
- Sachez que les informations d'identification du fichier partagé AWS `credentials` sont stockées en texte brut.

Conseils supplémentaires pour gérer les informations d'identification en toute sécurité

Pour avoir une idée générale de la façon de gérer en toute sécurité les informations d'identification AWS, consultez Informations d'identification de sécurité AWS dans le Références générales AWS et Bonnes pratiques de sécurité et cas d'utilisation dans le Guide de l'utilisateur IAM. En plus de ces exposés, envisagez de prendre les mesures suivantes :

- Créez des utilisateurs supplémentaires, tels que des utilisateurs dans IAM Identity Center, et utilisez leurs informations d'identification au lieu d'utiliser vos informations d'identification d'utilisateur root AWS. Les informations d'identification des autres utilisateurs peuvent être révoquées si nécessaire ou sont temporaires par nature. En outre, vous pouvez appliquer une politique à chaque utilisateur pour qu'il n'accède qu'à certaines ressources et actions et ainsi adopter les autorisations du moindre privilège.

- Utilisez des rôles IAM pour les tâches Amazon Elastic Container Service (Amazon ECS).

- Utilisez des rôles IAM pour les applications qui s'exécutent sur des instances Amazon EC2.

Rubriques

## Utilisation des informations d'identification AWS

Chaque commande des AWS Tools for PowerShell doit inclure un ensemble d'informations d'identification AWS, qui sont utilisées pour signer de manière chiffrée la demande de service web correspondante. Vous pouvez spécifier les informations d'identification par commande, par session ou pour toutes les sessions.

> **Warning**
>
> Afin d'éviter les risques de sécurité, n'employez pas les utilisateurs IAM pour l'authentification lorsque vous développez des logiciels spécialisés ou lorsque vous travaillez avec des données réelles. Préférez la fédération avec un fournisseur d'identité tel que AWS IAM Identity Center.

> **Note**
>
> Les informations contenues dans cette rubrique s'appliquent à des situations où vous devez obtenir et gérer manuellement des informations d'identification à court ou à long terme. Pour plus d'informations sur les informations d'identification à court et à long terme, consultez Other ways to authenticate dans le manuel AWS SDKs and Tools Reference Guide.
> Pour les bonnes pratiques en matière de sécurité, utilisez AWS IAM Identity Center comme décrit dans Configuration de l'authentification des outils.

En tant que bonne pratique, afin d'éviter d'exposer vos informations d'identification, ne placez pas d'informations d'identification littérales dans une commande. Au lieu de cela, créez un profil pour chaque ensemble d'informations d'identification que vous souhaitez utiliser et stockez le profil dans l'un ou l'autre des magasins d'informations d'identification. Spécifiez le profil approprié par son nom dans votre commande. Les AWS Tools for PowerShell récupéreront alors les informations d'identification associées. Pour obtenir une description générale de la manière de gérer en toute sécurité les informations d'identification AWS, veuillez consulter Bonnes pratiques en matière de gestion des clés d'accès AWS dans le Référence générale d'Amazon Web Services.

> **Note**
>
> Vous avez besoin d'un compte AWS pour obtenir des informations d'identification et utiliser les AWS Tools for PowerShell. Pour créer un compte AWS, consultez Getting started: Are you a first-time AWS user? dans le manuel AWS Account Management Reference Guide.

Rubriques



## Emplacements de stockage des informations d'identification

Les AWS Tools for PowerShell peuvent utiliser l'un ou l'autre des emplacements de stockage des informations d'identification :

- Le magasin SDK AWS, qui chiffre vos informations d'identification et les stocke dans votre dossier de base. Sous Windows, ce magasin se trouve à l'adresse suivante : `C:\Users\`*`username`*`\AppData\Local\AWSToolkit\RegisteredAccounts.json`.

  Les AWS SDK for .NET et Toolkit for Visual Studio peuvent également utiliser le magasin SDK AWS.

- Le fichier d'informations d'identification partagées, qui est également situé dans votre dossier de base, mais qui stocke les informations d'identification en texte brut.

  Par défaut, le fichier d'informations d'identification est stocké ici :

  - Sur Windows : `C:\Users\`*`username`*`\.aws\credentials`
  - Sur Mac/Linux : `~/.aws/credentials`

  Les kits SDK AWS et l'AWS Command Line Interface peuvent également utiliser le fichier d'informations d'identification. Si vous exécutez un script en dehors de votre contexte d'utilisateur AWS, veillez à ce que le fichier qui contient vos informations d'identification soit copié dans un

emplacement dans lequel tous les comptes d'utilisateur (système local et utilisateur) peuvent accéder à vos informations d'identification.

## Gestion des profils

Les profils vous permettent de référencer différents jeux d'informations d'identification avec les AWS Tools for PowerShell. Vous pouvez utiliser les applets de commande des AWS Tools for PowerShell pour gérer vos profils dans le magasin SDK AWS. Vous pouvez également gérer les profils du magasin SDK AWS à l'aide du Toolkit for Visual Studio ou par programmation à l'aide du AWS SDK for .NET. Pour obtenir des instructions de gestion des profils dans le fichier d'informations d'identification, veuillez consulter Bonnes pratiques en matière de gestion des clés d'accès AWS.

### Ajouter un profil

Pour ajouter un nouveau profil au magasin SDK AWS, exécutez la commande `Set-AWSCredential`. Le magasin stocke votre clé d'accès et votre clé secrète dans votre fichier d'informations d'identification par défaut sous le nom de profil que vous spécifiez.

```
PS > Set-AWSCredential `
                -AccessKey AKIA0123456787EXAMPLE `
                -SecretKey wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY `
                -StoreAs MyNewProfile
```

- `-AccessKey` – L'ID de la clé d'accès.
- `-SecretKey` – La clé secrète.
- `-StoreAs` – Le nom du profil, qui doit être unique. Pour spécifier le profil par défaut, utilisez le nom `default`.

### Mettre à jour un profil

Le magasin SDK AWS doit être tenu à jour manuellement. Si vous modifiez ultérieurement les informations d'identification du service, en utilisant par exemple la console IAM, l'exécution d'une commande avec les informations d'identification stockées localement échouera avec le message d'erreur suivant :

```
The Access Key Id you provided does not exist in our records.
```

Vous pouvez mettre à jour un profil en répétant la commande `Set-AWSCredential` pour le profil et en lui transmettant les nouvelles clé d'accès et clé secrète.

Répertorier les profils

Vous pouvez vérifier la liste actuelle des noms à l'aide de la commande suivante. Dans cet exemple, un utilisateur nommé Shirley a accès à trois profils qui sont tous stockés dans le fichier d'informations d'identification partagées (`~/.aws/credentials`).

```
PS > Get-AWSCredential -ListProfileDetail

ProfileName  StoreTypeName         ProfileLocation
-----------  -------------         ---------------
default      SharedCredentialsFile /Users/shirley/.aws/credentials
production   SharedCredentialsFile /Users/shirley/.aws/credentials
test         SharedCredentialsFile /Users/shirley/.aws/credentials
```

Supprimer un profil

Pour supprimer un profil dont vous n'avez plus besoin, utilisez la commande suivante.

```
PS > Remove-AWSCredentialProfile -ProfileName an-old-profile-I-do-not-need
```

Le paramètre `-ProfileName` spécifie le profil que vous souhaitez supprimer.

La commande obsolète Clear-AWSCredential est toujours disponible pour la rétrocompatibilité, mais vous devez lui préférer `Remove-AWSCredentialProfile`.

## Spécification des informations d'identification

Vous pouvez spécifier les informations d'identification de plusieurs façons. La méthode conseillée consiste à identifier un profil au lieu d'incorporer des informations d'identification littérales dans votre ligne de commande. Les AWS Tools for PowerShell localisent le profil à l'aide d'un ordre de recherche décrit dans Ordre de recherche des informations d'identification.

Sous Windows, les informations d'identification AWS stockées dans le magasin du kit SDK AWS sont chiffrées avec l'identité utilisateur Windows connectée. Elles ne peuvent pas être déchiffrées en utilisant un autre compte ou utilisées sur un appareil différent de celui sur lequel elles ont été créées à l'origine. Pour exécuter les tâches qui requièrent les informations d'identification d'un autre utilisateur, tel qu'un compte utilisateur sous lequel une tâche planifiée devra être exécutée, configurez un profil d'informations d'identification, comme décrit dans la section précédente, que vous

pourrez utiliser lorsque vous vous connecterez à l'ordinateur en tant qu'utilisateur. Connectez-vous en tant qu'utilisateur exécutant des tâches pour terminer les étapes de configuration des informations d'identification et créez un profil qui fonctionne pour cet utilisateur. Ensuite, déconnectez-vous, puis reconnectez-vous avec vos propres informations d'identification pour configurer la tâche planifiée.

> **Note**
>
> Utilisez le paramètre `-ProfileName` pour spécifier un profil. Ce paramètre est l'équivalent du paramètre `-StoredCredentials` des versions précédentes des AWS Tools for PowerShell. Pour des raisons de rétrocompatibilité, `-StoredCredentials` est toujours prise en charge.

Profil par défaut (recommandé)

Tous les kits SDK et les outils de gestion AWS peuvent trouver vos informations d'identification automatiquement sur votre ordinateur local si les informations d'identification sont stockées dans un profil nommé `default`. Par exemple, si vous avez un profil nommé `default` sur l'ordinateur local, vous n'avez pas à exécuter l'applet de commande `Initialize-AWSDefaultConfiguration` ou `Set-AWSCredential`. Les outils utilisent automatiquement les données de clé d'accès et de clé secrète stockées dans ce profil. Pour utiliser une région AWS autre que la région par défaut (les résultats de `Get-DefaultAWSRegion`), vous pouvez exécuter `Set-DefaultAWSRegion` et spécifier une région.

Si votre profil n'est pas nommé `default`, mais que vous souhaitez l'utiliser en tant que profil par défaut pour la session en cours, exécutez `Set-AWSCredential` pour le définir en tant que profil par défaut.

Même si l'exécution de `Initialize-AWSDefaultConfiguration` vous permet de spécifier un profil par défaut pour chaque session PowerShell, l'applet de commande charge les informations d'identification depuis votre profil dont le nom est personnalisé, mais remplace le profil `default` par le profil nommé.

Nous vous recommandons de ne pas exécuter, `Initialize-AWSDefaultConfiguration` sauf si vous exécutez une session PowerShell sur une instance Amazon EC2 qui n'a pas été lancée avec un profil d'instance, et que vous voulez configurer le profil d'informations d'identification manuellement. Notez que le profil d'informations d'identification ne contient pas d'informations d'identification dans ce scénario. Le profil d'informations d'identification résultant de l'exécution de `Initialize-AWSDefaultConfiguration` sur une instance EC2 ne stocke pas directement les informations

d'identification, mais pointe plutôt vers les métadonnées d'instance (qui fournissent des informations d'identification temporaires effectuant une rotation automatique). Cependant, il stocke la région de l'instance. Il se produit un autre scénario pouvant nécessiter l'exécution de `Initialize-AWSDefaultConfiguration` si vous voulez exécuter un appel sur une région autre que celle dans laquelle l'instance s'exécute. L'exécution de cette commande remplace définitivement la région stockée dans les métadonnées de l'instance.

```
PS > Initialize-AWSDefaultConfiguration -ProfileName MyProfileName -Region us-west-2
```

> **Note**
>
> Par défaut, les informations d'identification sont incluses dans le magasin SDK AWS sous le nom de profil `default`. La commande remplace tout profil existant par ce nom.

Si votre instance EC2 a été lancée avec un profil d'instance, PowerShell obtient automatiquement les informations d'identification AWS et les informations de région depuis le profil d'instance. Vous n'avez pas besoin d'exécuter `Initialize-AWSDefaultConfiguration`. L'exécution de l'applet de commande `Initialize-AWSDefaultConfiguration` sur une instance EC2 lancée avec un profil d'instance n'est pas nécessaire, car l'applet utilise les mêmes données de profil d'instance que celles déjà utilisées par PowerShell par défaut.

Profil de session

Utilisez `Set-AWSCredential` pour spécifier un profil par défaut pour une session particulière. Ce profil remplace tout profil par défaut pendant la durée de la session. Nous vous recommandons de l'utiliser si vous souhaitez utiliser un profil dont le nom est personnalisé dans votre session au lieu du profil `default` actuel.

```
PS > Set-AWSCredential -ProfileName MyProfileName
```

> **Note**
>
> Dans les versions des Tools for Windows PowerShell antérieures à 1.1, l'applet de commande `Set-AWSCredential` ne fonctionnait pas correctement et vous deviez remplacer le profil spécifié par « MyProfileName ». Nous vous recommandons d'utiliser une version plus récente des Tools for Windows PowerShell.

Profil de commande

Dans les commandes individuelles, vous pouvez ajouter le paramètre `-ProfileName` pour spécifier un profil qui ne s'applique qu'à cette commande. Ce profil remplace tous les profils par défaut ou de session, comme illustré dans l'exemple suivant.

```
PS > Get-EC2Instance -ProfileName MyProfileName
```

> **Note**
>
> Lorsque vous indiquez un profil de session ou un profil par défaut, vous pouvez également ajouter un paramètre `-Region` pour remplacer une région de session ou par défaut. Pour de plus amples informations, veuillez consulter Spécifier AWS les régions. L'exemple suivant spécifie un profil et une région par défaut.
>
> ```
> PS > Initialize-AWSDefaultConfiguration -ProfileName MyProfileName -Region us-
> west-2
> ```

Par défaut, le fichier d'informations d'identification partagées AWS est supposé être dans le dossier de base de l'utilisateur (`C:\Users\username\.aws` sous Windows ou `~/.aws` sous Linux). Pour spécifier un fichier d'informations d'identification situé dans un autre emplacement, incluez le paramètre `-ProfileLocation` et spécifiez le chemin d'accès du fichier d'informations d'identification. L'exemple suivant spécifie un fichier d'informations d'identification autre que par défaut pour une commande spécifique.

```
PS > Get-EC2Instance -ProfileName MyProfileName -ProfileLocation C:
\aws_service_credentials\credentials
```

> **Note**
>
> Si vous exécutez un script PowerShell à un moment où vous n'êtes pas normalement connecté à AWS (par exemple, vous exécutez un script PowerShell comme une tâche planifiée en dehors de vos heures de travail habituelles), ajoutez le paramètre `-ProfileLocation` lorsque vous spécifiez le profil que vous souhaitez utiliser et définissez la valeur du chemin du fichier qui stocke vos informations d'identification. Pour être certain que votre script des AWS Tools for PowerShell s'exécute avec les informations d'identification

de compte correctes, vous devez ajouter le paramètre `-ProfileLocation` chaque fois que votre script s'exécute dans un contexte ou processus qui n'utilise pas de compte AWS. Vous pouvez également copier votre fichier d'informations d'identification dans un emplacement accessible au système local ou dans un autre compte que vos scripts utilisent pour exécuter les tâches.

## Ordre de recherche des informations d'identification

Lorsque vous exécutez une commande, les AWS Tools for PowerShell recherchent les informations d'identification dans l'ordre suivant. Ils s'arrêtent lorsqu'ils trouvent des informations d'identification utilisables.

1. Informations d'identification littérales intégrées en tant que paramètres dans la ligne de commande.

   Nous vous recommandons vivement d'utiliser des profils plutôt que de placer des informations d'identification littérales dans vos lignes de commande.
2. Un nom de profil ou un emplacement de profil spécifié.
   - Si vous spécifiez uniquement un nom de profil, la commande recherchera le profil spécifié dans le magasin SDK AWS et, s'il n'existe pas, le profil spécifié dans le fichier d'informations d'identification partagées AWS de l'emplacement par défaut.
   - Si vous spécifiez uniquement un emplacement de profil, la commande recherchera le profil `default` dans ce fichier d'informations d'identification.
   - Si vous spécifiez à la fois un nom et un emplacement, la commande recherchera le profil spécifié dans ce fichier d'informations d'identification.

   Si le profil ou l'emplacement spécifié est introuvable, la commande lèvera une exception. La recherche exécute les étapes suivantes uniquement si vous n'avez pas spécifié de profil ou d'emplacement.
3. Informations d'identification spécifiées par le paramètre `-Credential`.
4. Le profil de session, s'il en existe un.
5. Le profil par défaut, dans l'ordre suivant :
   a. Le profil `default` dans le magasin SDK AWS.
   b. Le profil `default` dans le fichier d'informations d'identification partagées AWS.
   c. Le profil `AWS PS Default` dans le magasin SDK AWS.

6. Si la commande s'exécute sur une instance Amazon EC2 configurée pour utiliser un rôle IAM, les informations d'identification temporaires de l'instance EC2 auxquelles le système a accédé à partir du profil d'instance.

   Pour en savoir plus sur l'utilisation des rôles IAM pour les instances Amazon EC2, consultez la section AWS SDK for .NET.

Si cette recherche échoue à retrouver les informations d'identification spécifiées, la commande lèvera une exception.

## Gestion des informations d'identification dans les AWS Tools for PowerShell Core

Les applets de commande des AWS Tools for PowerShell Core acceptent les clés d'accès et les clés secrètes AWS ou les noms des profils d'informations d'identification lorsqu'ils s'exécutent, de la même manière que les AWS Tools for Windows PowerShell. Lorsqu'ils s'exécutent sur Windows, les deux modules ont accès au fichier du magasin d'informations d'identification du kit AWS SDK for .NET (stocké dans le fichier `AppData\Local\AWSToolkit\RegisteredAccounts.json` par utilisateur).

Ce fichier stocke vos clés dans un format chiffré et ne peut pas être utilisé sur un autre ordinateur. Il s'agit du premier fichier dans lequel les AWS Tools for PowerShell recherchent un profil d'informations d'identification et il s'agit également du fichier dans lequel les AWS Tools for PowerShell stockent les profils d'informations d'identification. Pour de plus amples informations sur le fichier du magasin d'informations d'identification du AWS SDK for .NET, consultez Configuration des informations d'identification AWS. Le module Tools for Windows PowerShell ne prend pas en charge actuellement l'écriture des informations d'identification sur d'autres fichiers ou emplacements.

Ces deux modules peuvent lire les profils à partir du fichier d'informations d'identification partagées AWS, lequel fichier est utilisé par d'autres kits SDK AWS et par l'AWS CLI. Sous Windows, l'emplacement par défaut du fichier est `C:\Users\<userid>\.aws\credentials`. Sur les plates-formes autres que Windows, le fichier est stocké à l'emplacement `~/.aws/credentials`. Le paramètre `-ProfileLocation` peut être utilisé pour pointer vers un nom de fichier ou emplacement du fichier autre que par défaut.

Le magasin d'informations d'identification du kit SDK conserve vos informations d'identification sous une forme chiffrée à l'aide des API de chiffrement Windows. Ces API ne sont pas disponibles sur d'autres plates-formes, de telle sorte que le module des AWS Tools for PowerShell Core utilise exclusivement le fichier d'informations d'identification partagées AWS et prend en charge l'écriture de nouveaux profils d'informations d'identification dans le fichier d'informations d'identification partagées.

Les exemples de scripts suivants qui utilisent l'applet de commande `Set-AWSCredential` montrent les options de gestion des profils d'informations d'identification sous Windows avec le module AWSPowerShell ou le module AWSPowerShell.NetCore.

```
# Writes a new (or updates existing) profile with name "myProfileName"
# in the encrypted SDK store file

Set-AWSCredential -AccessKey akey -SecretKey skey -StoreAs myProfileName

# Checks the encrypted SDK credential store for the profile and then
# falls back to the shared credentials file in the default location

Set-AWSCredential -ProfileName myProfileName

# Bypasses the encrypted SDK credential store and attempts to load the
# profile from the ini-format credentials file "mycredentials" in the
# folder C:\MyCustomPath

Set-AWSCredential -ProfileName myProfileName -ProfileLocation C:\MyCustomPath
\mycredentials
```

Les exemples suivants illustrent le comportement du module AWSPowerShell.NetCore sous Linux ou macOS.

```
# Writes a new (or updates existing) profile with name "myProfileName"
# in the default shared credentials file ~/.aws/credentials

Set-AWSCredential -AccessKey akey -SecretKey skey -StoreAs myProfileName

# Writes a new (or updates existing) profile with name "myProfileName"
# into an ini-format credentials file "~/mycustompath/mycredentials"

Set-AWSCredential -AccessKey akey -SecretKey skey -StoreAs myProfileName -
ProfileLocation ~/mycustompath/mycredentials

# Reads the default shared credential file looking for the profile "myProfileName"

Set-AWSCredential -ProfileName myProfileName

# Reads the specified credential file looking for the profile "myProfileName"
```

```
Set-AWSCredential -ProfileName myProfileName -ProfileLocation ~/mycustompath/
mycredentials
```

## Informations d'identification partagées dans AWS Tools for PowerShell

Les Tools for Windows PowerShell prennent en charge l'utilisation du fichier d'informations d'identification partagées AWS, de la même manière que l'AWS CLI et les autres kits SDK AWS. Les Tools for Windows PowerShell prennent désormais en charge la lecture et l'écriture de profils d'informations d'identification `basic`, `session` et `assume role` dans le fichier d'informations d'identification .NET et le fichier d'informations d'identification partagées AWS. Cette fonctionnalité est activée par un nouvel espace de noms `Amazon.Runtime.CredentialManagement`.

> **Warning**
>
> Afin d'éviter les risques de sécurité, n'employez pas les utilisateurs IAM pour l'authentification lorsque vous développez des logiciels spécialisés ou lorsque vous travaillez avec des données réelles. Préférez la fédération avec un fournisseur d'identité tel que AWS IAM Identity Center.

> **Note**
>
> Les informations contenues dans cette rubrique s'appliquent à des situations où vous devez obtenir et gérer manuellement des informations d'identification à court ou à long terme. Pour plus d'informations sur les informations d'identification à court et à long terme, consultez Other ways to authenticate dans le manuel AWS SDKs and Tools Reference Guide.
> Pour les bonnes pratiques en matière de sécurité, utilisez AWS IAM Identity Center comme décrit dans Configuration de l'authentification des outils.

Les nouveaux types de profil et l'accès au fichier d'informations d'identification partagées AWS sont pris en charge par les paramètres suivants, qui ont été ajoutés aux applets de commande relatives aux informations d'identification, Initialize-AWSDefaultConfiguration, New-AWSCredential et Set-AWSCredential. Dans les applets de commande des services, vous pouvez faire référence à vos profils en ajoutant le paramètre commun, `-ProfileName`.

## Utilisation d'un rôle IAM avec AWS Tools for PowerShell

Le fichier d'informations d'identification partagées AWS permet d'autres types d'accès. Par exemple, vous pouvez accéder à vos ressources AWS en utilisant un rôle IAM au lieu des informations d'identification à long terme d'un utilisateur IAM. Pour ce faire, vous devez posséder un profil standard disposant des autorisations nécessaires pour assumer le rôle. Lorsque vous demandez aux AWS Tools for PowerShell d'utiliser un profil qui a spécifié un rôle, les AWS Tools for PowerShell recherchent le profil identifié par le biais du paramètre `SourceProfile`. Ces informations d'identification sont utilisées pour demander des informations d'identification temporaires pour le rôle spécifié par le paramètre `RoleArn`. Vous pouvez éventuellement exiger l'utilisation d'un appareil MFA (Multi-Factor Authentication, authentification multifacteur) ou d'un code `ExternalId` lorsque le rôle est assumé par un tiers.

| Nom du paramètre | Description |
| --- | --- |
| ExternalId | ID externe défini par l'utilisateur à utiliser lors de l'endossement d'un rôle, si requis par le rôle. Cela n'est généralement requis que lorsque vous déléguez l'accès à votre compte à un tiers. Le tiers doit inclure l'ExternalId comme paramètre lorsqu'il assume le rôle assigné. Pour plus d'informations, consultez Procédure d'utilisation d'un ID externe lorsque vous accordez l'accès à vos ressources AWS à un tiers dans le guide de l'utilisateur IAM. |
| MfaSerial | Numéro de série MFA à utiliser lors de l'endossement d'un rôle, si requis par le rôle. Pour de plus amples informations, veuillez consulter Utilisation de l'Authentification multi-facteur (MFA) dans AWS dans le guide de l'utilisateur IAM. |
| RoleArn | ARN du rôle à endosser pour endosser les informations d'identification du rôle. Pour plus d'informations sur la création et l'utilisation des |

| Nom du paramètre | Description |
| --- | --- |
| | rôles IAM, consultez Rôles IAM dans le guide de l'utilisateur IAM. |
| SourceProfile | Nom du profil de la source à utiliser par les informations d'identification du rôle responsable. Les informations d'identification trouvées dans ce profil sont utilisées pour assumer le rôle spécifié par le paramètre `RoleArn`. |

Configuration des profils pour assumer un rôle

Voici un exemple montrant comment configurer un profil source permettant d'assumer directement un rôle IAM.

La première commande crée un profil source référencé par le profil de rôle. La deuxième commande crée le profil de rôle sur la base duquel le rôle sera assumé. La troisième commande affiche les informations d'identification du profil de rôle.

```
PS > Set-AWSCredential -StoreAs my_source_profile -AccessKey access_key_id -
SecretKey secret_key
PS > Set-AWSCredential -StoreAs my_role_profile -SourceProfile my_source_profile -
RoleArn arn:aws:iam::123456789012:role/role-i-want-to-assume
PS > Get-AWSCredential -ProfileName my_role_profile

SourceCredentials                  RoleArn
 RoleSessionName                          Options
-----------------                  -------
 ---------------                          -------
Amazon.Runtime.BasicAWSCredentials arn:aws:iam::123456789012:role/
role-i-want-to-assume aws-dotnet-sdk-session-636238288466144357
 Amazon.Runtime.AssumeRoleAWSCredentialsOptions
```

Pour utiliser ce profil de rôle avec les applets de commande de service des Tools for Windows PowerShell, ajoutez le paramètre commun `-ProfileName` à la commande pour référencer le profil de rôle. L'exemple suivant utilise le profil de rôle défini dans l'exemple précédent pour accéder à l'applet de commande Get-S3Bucket. Les AWS Tools for PowerShell recherchent les informations d'identification dans `my_source_profile`, utilisent ces informations d'identification

pour appeler `AssumeRole` au nom de l'utilisateur, puis utilisent ces informations d'identification de rôle temporaires pour appeler `Get-S3Bucket`.

```
PS > Get-S3Bucket -ProfileName my_role_profile

CreationDate           BucketName
------------           ----------
2/27/2017 8:57:53 AM   4ba3578c-f88f-4d8b-b95f-92a8858dac58-bucket1
2/27/2017 10:44:37 AM  2091a504-66a9-4d69-8981-aaef812a02c3-bucket2
```

## Utilisation des types de profil d'informations d'identification

Pour définir un type de profil d'informations d'identification, déterminez les paramètres qui fournissent les informations requises par le type de profil.

| Type d'informations d'identification | Paramètres que vous devez utiliser |
|---|---|
| Base<br><br>Ce sont les informations d'identification à long terme d'un utilisateur IAM | `-AccessKey`<br><br>`-SecretKey` |
| Session :<br><br>Il s'agit des informations d'identification à court terme d'un rôle IAM que vous récupérez manuellement, par exemple en appelant directement l'applet de commande Use-STSRole. | `-AccessKey`<br><br>`-SecretKey`<br><br>`-SessionToken` |
| Rôle :<br><br>Il s'agit d'informations d'identification à court terme d'un rôle IAM que les AWS Tools for PowerShell récupèrent pour vous. | `-SourceProfile`<br><br>`-RoleArn`<br><br>facultatif : `-ExternalId`<br><br>facultatif : `-MfaSerial` |

## Paramètre commun **`ProfilesLocation`**

Vous pouvez utiliser `-ProfileLocation` pour écrire dans le fichier d'informations d'identification partagées, ainsi que pour demander à une applet de commande de lire à partir du fichier d'informations d'identification. L'ajout du paramètre `-ProfileLocation` détermine si les Tools for Windows PowerShell utilisent le fichier d'informations d'identification partagées ou le fichier d'informations d'identification .NET. Le tableau suivant décrit comment le paramètres fonctionne dans les Tools for Windows PowerShell.

| Valeur de l'emplacement du profil | Comportement de résolution du profil |
|---|---|
| null (non défini) ou vide | Tout d'abord, recherchez dans le fichier d'informations d'identification .NET un profil portant le nom spécifié. Si le profil est introuvable, recherchez le fichier d'informations d'identification partagées AWS à l'adresse *(user's home directory)* `\.aws\credentials` . |
| Chemin d'accès d'un fichier dans le format de fichier d'informations d'identification partagées AWS | Recherchez un profil portant le nom indiqué uniquement dans le fichier spécifié. |

Enregistrement des informations d'identification dans un fichier d'informations d'identification

Pour écrire et enregistrer les informations d'identification dans l'un des deux fichiers d'informations d'identification, exécutez l'applet de commande `Set-AWSCredential`. L'exemple suivant illustre la marche à suivre. La première commande utilise `Set-AWSCredential` avec `-ProfileLocation` pour ajouter des clés d'accès et des clés secrètes à un profil spécifié par le paramètre `-ProfileName`. Dans la deuxième ligne, exécutez l'applet de commande Get-Content pour afficher le contenu du fichier d'informations d'identification.

```
PS > Set-AWSCredential -ProfileLocation C:\Users\auser\.aws\credentials -ProfileName
 basic_profile -AccessKey access_key2 -SecretKey secret_key2
PS > Get-Content C:\Users\auser\.aws\credentials

aws_access_key_id=access_key2
aws_secret_access_key=secret_key2
```

## Affichage de vos profils d'informations d'identification

Exécutez l'applet de commande Get-AWSCredential et ajoutez le paramètre `-ListProfileDetail` pour renvoyer les types et les emplacements des fichiers d'informations d'identification, ainsi qu'une liste des noms de profils.

```
PS > Get-AWSCredential -ListProfileDetail

ProfileName                   StoreTypeName         ProfileLocation
-----------                   -------------         ---------------
source_profile                NetSDKCredentialsFile
assume_role_profile           NetSDKCredentialsFile
basic_profile                 SharedCredentialsFile C:\Users\auser\.aws\credentials
```

## Suppression des profils d'informations d'identification

Pour supprimer les profils d'informations d'identification, exécutez la nouvelle applet de commande Remove-AWSCredentialProfile. Clear-AWSCredential est obsolète, mais est toujours disponible pour la rétrocompatibilité.

## Remarques importantes

Seules les applets de commande Initialize-AWSDefaultConfiguration, New-AWSCredential et Set-AWSCredential prennent en charge les paramètres des profils de rôle. Vous ne pouvez pas spécifier les paramètres de rôle directement dans une commande telle que `Get-S3Bucket -SourceProfile source_profile_name -RoleArn arn:aws:iam::999999999999:role/role_name`. Cela ne fonctionne pas, car les applets de commande de service ne prennent pas directement en charge les paramètres `SourceProfile` ou `RoleArn`. Au lieu de cela, vous devez stocker ces paramètres dans un profil, puis appeler la commande avec le paramètre `-ProfileName`.

# Caractéristiques du AWS Tools for Windows PowerShell

Certaines rubriques de cette section fournissent des informations sur les fonctionnalités des outils pour Windows PowerShell que vous devrez peut-être prendre en compte lors de la création de vos projets et de vos scripts. D'autres rubriques de cette section fournissent des informations sur les méthodes avancées de configuration des outils, de votre environnement et de vos projets. Assurez-vous d'avoir d'abord installé et configuré les outils.

Pour plus d'informations sur le développement de logiciels pour des AWS services spécifiques ainsi que des exemples de code, consultezTravaillez avec les AWS services. Pour des exemples de code supplémentaires, voirOutils pour des exemples PowerShell de code.

Rubriques



## Observabilité

> Ceci est une documentation préliminaire pour une version préliminaire de service. Elle est susceptible d'être modifiée.

L'observabilité est la mesure dans laquelle l'état actuel d'un système peut être déduit des données qu'il émet. Les données émises sont communément appelées télémétrie. Pour plus d'informations sur la télémétrie lors de l'utilisation de AWS services, consultez la section Observabilité dans le guide du AWS SDK for .NET développeur.

Le code suivant montre un exemple de la manière dont l'observabilité peut être activée dans le AWS Tools for PowerShell.

```
<#
    This is an example of generating telemetry for AWS Tools for PowerShell.
    Each cmdlet that interacts with an Amazon Web Service creates a trace containing
 spans
    for underlying processes and AWS SDK for .NET operations.
    This example is written using PowerShell 7 and .NET 8.
    It requires the installation of the .NET CLI tool.
    Note that implementation varies by the exporter/endpoint, which is not specified in
 this example.
```

```
    For more information, see https://opentelemetry.io/docs/languages/net/exporters/.
#>

# Set this value to a common folder path on your computer for local development of code
 repositories.
$devProjectsPath = [System.IO.Path]::Join('C:', 'Dev', 'Repos')

# If these values are changed, update the hardcoded method invocation toward the end of
 this script.
# Values must follow constraints for namespaces and classes.
$telemetryProjectName = 'ExampleAWSPowerShellTelemetryImplementation'
$serviceName = 'ExamplePowerShellService'

# This example supposes that the OTLP exporter requires these two properties,
# but some exporters require different properties or no properties.
$telemetryEndPoint = 'https://example-endpoint-provider.io'
$telemetryHeaders = 'x-example-header=abc123'

$dllsPath = [System.IO.Path]::Join($devProjectsPath, $telemetryProjectName, 'bin',
 'Release', 'net8.0', 'publish')

$telemetryProjectPath = [System.IO.Path]::Join($devProjectsPath, $telemetryProjectName)

# This script is designed to recreate the example telemetry project each time it's
 executed.
Remove-Item -Path $telemetryProjectPath -Recurse -Force -ErrorAction 'SilentlyContinue'
$null = New-Item -Path $devProjectsPath -Name $telemetryProjectName -ItemType
 'Directory'

<#
    Create and build a C#-based .NET 8 project that implements
    OpenTelemetry Instrumentation for the AWS Tools for PowerShell.
#>

Set-Location -Path $telemetryProjectPath

dotnet new classlib

# Other exporters are available.
# For more information, see https://opentelemetry.io/docs/languages/net/exporters/.
dotnet add package OpenTelemetry.Exporter.OpenTelemetryProtocol
dotnet add package OpenTelemetry.Instrumentation.AWS --prerelease

$classContent = @"
```

```
using OpenTelemetry;
using OpenTelemetry.Resources;
using OpenTelemetry.Trace;

namespace Example.Telemetry;

public class AWSToolsForPowerShellTelemetry
{
    public static void InitializeAWSInstrumentation()
    {
        Sdk.CreateTracerProviderBuilder()
        .ConfigureResource(e => e.AddService("$ServiceName"))
        .AddAWSInstrumentation()
        // Exporters vary so options might need to be changed or omitted.
        .AddOtlpExporter(options =>
        {
            options.Endpoint = new Uri("$telemetryEndPoint");
            options.Headers = "$telemetryHeaders";
        })
        .Build();
    }
}
"@

$csFilePath = [System.IO.Path]::Join($telemetryProjectPath, ($serviceName + '.cs'))
Set-Content -Path $csFilePath -Value $classContent

dotnet build
dotnet publish -c Release

<#
    Add additional modules here for any other cmdlets that you require.
    Beyond this point, additional AWS Tools for PowerShell modules will fail to import
    due to conflicts with the AWS SDK for .NET assemblies that are added next.
#>

Import-Module -Name 'AWS.Tools.Common'
Import-Module -Name 'AWS.Tools.DynamoDBv2'


# Load assemblies for the telemetry project, excluding the AWS SDK for .NET assemblies
# that were already loaded by importing AWS Tools for PowerShell modules.
$dlls = (Get-ChildItem $dllsPath -Filter *.dll -Recurse ).FullName
```

```
$AWSSDKAssembliesAlreadyLoaded =
 [Threading.Thread]::GetDomain().GetAssemblies().Location | Where-Object {$_ -like
 '*AWSSDK*' } | Split-Path -Leaf
$dlls.Where{$AWSSDKAssembliesAlreadyLoaded -notcontains ($_ | Split-Path -
Leaf)}.ForEach{Add-Type -Path $_}

# Invoke the method defined earlier in this script.
[Example.Telemetry.AWSToolsForPowerShellTelemetry]::InitializeAWSInstrumentation()

<#
    Now telemetry will be exported for AWS Tools for PowerShell cmdlets
    that are invoked directly or indirectly.
    Execute this cmdlet or execute your own PowerShell script.
#>
Get-DDBTable -TableName 'DotNetTests-HashTable' -Region 'us-east-1'
```

# Travaillez avec AWS les services du AWS Tools for PowerShell

Cette section fournit des exemples d'utilisation des services AWS Tools for PowerShell pour accéder aux AWS services. Ces exemples aident à montrer comment utiliser les applets de commande pour effectuer des tâches réelles AWS . Ces exemples s'appuient sur des applets de commande fournis par Tools for PowerShell . Pour déterminer les applets de commande disponibles, consultez la référence des applets de commande AWS Tools for PowerShell.

## PowerShell Codage par concaténation de fichiers

Certaines applets de commande permettent de AWS Tools for PowerShell modifier des fichiers ou des enregistrements existants que vous avez. AWS Par exemple`Edit-R53ResourceRecordSet`, qui appelle Amazon Route 53. ChangeResourceRecordSetsAPI

Lorsque vous modifiez ou concaténez des fichiers dans les versions PowerShell 5.1 ou antérieures, PowerShell encode la sortie en -16, et non en UTF -8. UTF Cela peut ajouter des caractères indésirables et créer des résultats non valides. Un éditeur hexadécimal peut afficher les caractères indésirables.

Pour éviter de convertir la sortie du fichier en UTF -16, vous pouvez diriger votre commande vers PowerShell l'`Out-File`applet de commande et spécifier le codage UTF -8, comme illustré dans l'exemple suivant :

```
PS > *some file concatenation command* | Out-File filename.txt -Encoding utf8
```

Si vous exécutez des AWS CLI commandes depuis la PowerShell console, le même comportement s'applique. Vous pouvez rediriger le résultat d'une AWS CLI commande vers `Out-File` la PowerShell console. D'autres applets de commande, comme `Export-Csv` ou `Export-Clixml`, disposent également d'un paramètre `Encoding`. Pour obtenir la liste complète des applets de commande ayant un paramètre `Encoding` et permettant de corriger l'encodage de la sortie d'un fichier concaténé, exécutez la commande suivante :

```
PS > Get-Command -ParameterName "Encoding"
```

> **Note**
>
> PowerShell Les versions 6.0 et ultérieures, y compris PowerShell Core, conservent automatiquement le codage UTF -8 pour la sortie de fichiers concaténés.

# Objets renvoyés pour les PowerShell outils

Pour être AWS Tools for PowerShell plus utile dans un PowerShell environnement natif, l'objet renvoyé par une AWS Tools for PowerShell applet de commande est un. NET, et non l'objet JSON texte qui est généralement renvoyé par le correspondant API dans le AWS SDK. Par exemple, `Get-S3Bucket` émet une `Buckets` collection et non un objet de JSON réponse Amazon S3. La `Buckets` collection peut être placée dans le PowerShell pipeline et interagir avec elle de manière appropriée. De même, `Get-EC2Instance` émet un`Reservation`. NETcollection d'objets, pas un objet de `DescribeEC2Instances` JSON résultat. Ce comportement est intentionnel et permet à l' AWS Tools for PowerShell expérience d'être plus cohérente avec l'idiomatique PowerShell.

Les réponses de service réelles sont à votre disposition si vous en avez besoin. Elles sont stockées en tant que propriétés `note` dans les objets renvoyés. Pour les API actions qui prennent en charge la pagination à l'aide de `NextToken` champs, ceux-ci sont également attachés en tant que `note` propriétés.

## Amazon EC2

Cette section décrit les étapes requises pour lancer une EC2 instance Amazon, notamment comment :

- Récupérez une liste d'Amazon Machine Images (AMIs).
- Créez une paire de clés pour SSH l'authentification.
- Créez et configurez un groupe EC2 de sécurité Amazon.
- Lancer l'instance et récupérer les informations associées.

## Amazon S3

Cette section présente les étapes nécessaires pour créer un site Web statique hébergé dans Amazon S3. Elle explique comment :

- Créer et supprimer des compartiments Amazon S3.
- Charger des fichiers dans un compartiment Amazon S3 en tant qu'objets.
- Supprimer des objets d'un compartiment Amazon S3.
- Désigner un compartiment Amazon S3 en tant que site Web.

## AWS Lambda et AWS Tools for PowerShell

Cette section fournit un bref aperçu des outils AWS Lambda pour le PowerShell module et décrit les étapes requises pour configurer le module.

## Amazon SNS et Amazon SQS

Cette section décrit les étapes nécessaires pour abonner une SQS file d'attente Amazon à un SNS sujet Amazon. Elle explique comment :

- Créez un SNS sujet Amazon.
- Créez une SQS file d'attente Amazon.
- Abonnez la file d'attente à la rubrique .
- Envoyer un message à la rubrique.
- Recevoir le message de la file d'attente.

## CloudWatch

Cette section fournit un exemple de publication de données personnalisées sur CloudWatch.

- Publiez une métrique personnalisée sur votre CloudWatch tableau de bord.

## consultez aussi

- Premiers pas avec AWS Tools for Windows PowerShell

## Rubriques

- Amazon S3 et Tools for Windows PowerShell

- Amazon EC2 et Tools for Windows PowerShell
- AWS Lambda et AWS Tools for PowerShell
- Amazon SQS, Amazon SNS et Tools for Windows PowerShell
- CloudWatch depuis le AWS Tools for Windows PowerShell
- Utilisation du paramètre ClientConfig dans les applets de commande

# Amazon S3 et Tools for Windows PowerShell

Dans cette section, nous créons un site Web statique à l'aide AWS Tools for Windows PowerShell de utilisant Amazon S3 et CloudFront. Dans le processus, nous proposons un certain nombre de tâches courantes avec ces services. Cette procédure est modélisée d'après la procédure d'hébergement d'un site Web statique du Guide de démarrage, qui décrit un processus similaire en utilisant la AWSManagement Console.

Les commandes présentées ici supposent que vous avez défini des informations d'identification par défaut et une région par défaut pour votre session PowerShell. Par conséquent, les informations d'identification et les régions ne sont pas incluses dans l'appel des applets de commande.

> **Note**
>
> Il n'existe actuellement aucune API Amazon S3 permettant de renommer un compartiment ou un objet, et donc aucune applet de commande unique Tools for Windows PowerShell permettant d'effectuer cette tâche. Pour renommer un objet dans S3, nous vous recommandons de le copier dans un autre objet sous un nouveau nom en exécutant l'applet de commande Copy-S3Object, puis de supprimer l'objet d'origine en exécutant l'applet de commande Remove-S3Object.

Consulter aussi

- Travaillez avec AWS les services du AWS Tools for PowerShell
- Hébergement d'un site web statique sur Amazon S3
- Console Amazon S3

Rubriques

- Créer un compartiment Amazon S3, vérifier sa région et le supprimer (facultatif)

- Configuration d'un compartiment Amazon S3 comme site Web et activation de la journalisation
- Charger les objets sur un compartiment Amazon S3
- Suppression d'objets et de compartiments Amazon S3
- Charger du contenu du texte en ligne sur Amazon S3

## Créer un compartiment Amazon S3, vérifier sa région et le supprimer (facultatif)

Utilisez l'applet de commande `New-S3Bucket` pour créer un nouveau compartiment Amazon S3. Les exemples suivants créent un compartiment nommé `website-example`. Le nom du compartiment doit être unique dans toutes les régions. L'exemple montre comment créer le compartiment dans la région `us-west-1`.

```
PS > New-S3Bucket -BucketName website-example -Region us-west-2

CreationDate         BucketName
------------         ----------
8/16/19 8:45:38 PM   website-example
```

Vous pouvez vérifier la région dans laquelle le compartiment se trouve à l'aide de l'applet de commande `Get-S3BucketLocation`.

```
PS > Get-S3BucketLocation -BucketName website-example

Value
-----
us-west-2
```

Une fois que vous avez terminé ce didacticiel, vous pouvez utiliser la ligne suivante pour supprimer ce compartiment. Nous vous conseillons de laisser ce compartiment en place, car nous l'utiliserons dans de prochains exemples.

```
PS > Remove-S3Bucket -BucketName website-example
```

Notez que le processus de suppression du compartiment peut prendre un certain temps. Si vous essayez de recréer immédiatement un compartiment du même nom, l'applet de commande `New-S3Bucket` peut échouer tant que l'ancien compartiment n'a pas complètement disparu.

### Voir aussi

- Travaillez avec AWS les services du AWS Tools for PowerShell
- Put Bucket (Référence des services Amazon S3)
- AWSRégions PowerShell pour Amazon S3

## Configuration d'un compartiment Amazon S3 comme site Web et activation de la journalisation

Utilisez l'applet de commande `Write-S3BucketWebsite` pour configurer un compartiment Amazon S3 comme site Web statique. L'exemple suivant spécifie le nom `index.html` pour la page Web de contenu par défaut et le nom `error.html` pour la page Web d'erreur par défaut. Notez que l'applet de commande ne crée pas ces pages. Elles doivent être chargées en tant qu'objets Amazon S3.

```
PS > Write-S3BucketWebsite -BucketName website-example -
WebsiteConfiguration_IndexDocumentSuffix index.html -WebsiteConfiguration_ErrorDocument
 error.html
RequestId      : A1813E27995FFDDD
AmazonId2      : T7hlDOeLqA5Q2XfTe8j2q3SLoP3/5XwhUU3RyJBGHU/LnC+CIWLeGgP0MY24xAlI
ResponseStream :
Headers        : {x-amz-id-2, x-amz-request-id, Content-Length, Date...}
Metadata       : {}
ResponseXml    :
```

### Voir aussi

- Travaillez avec AWS les services du AWS Tools for PowerShell
- Autorisation PutBucketWebsite (Référence d'API Amazon S)
- Autorisation PutBucketACL (Référence d'API Amazon S)

## Charger les objets sur un compartiment Amazon S3

Utilisez l'applet de commande `Write-S3Object` pour charger des fichiers de votre système de fichiers local dans un compartiment Amazon S3 en tant qu'objets. L'exemple ci-dessous crée et charge deux fichiers HTML simples dans un compartiment Amazon S3 et vérifie l'existence des objets chargés. Le paramètre `-File` de `Write-S3Object` spécifie le nom du fichier dans le

système de fichiers local. Le paramètre `-Key` spécifie le nom que l'objet correspondant aura dans Amazon S3.

Amazon déduit le type de contenu des objets à partir des extensions de fichier : « .html », dans le cas présent.

```
PS > # Create the two files using here-strings and the Set-Content cmdlet
PS > $index_html = @"
>> <html>
>>   <body>
>>     <p>
>>       Hello, World!
>>     </p>
>>   </body>
>> </html>
>> "@
>>
PS > $index_html | Set-Content index.html
PS > $error_html = @"
>> <html>
>>   <body>
>>     <p>
>>       This is an error page.
>>     </p>
>>   </body>
>> </html>
>> "@
>>
>>$error_html | Set-Content error.html
>># Upload the files to Amazon S3 using a foreach loop
>>foreach ($f in "index.html", "error.html") {
>> Write-S3Object -BucketName website-example -File $f -Key $f -CannedACLName public-
read
>> }
>>
PS > # Verify that the files were uploaded
PS > Get-S3BucketWebsite -BucketName website-example

IndexDocumentSuffix                                      ErrorDocument
-------------------                                      -------------
index.html                                               error.html
```

Options ACL prêtes à l'emploi

Les valeurs pour spécifier les listes de contrôle d'accès (ACL) prêtes à l'emploi avec les Tools for Windows PowerShell sont les mêmes que celles utilisées par le AWS SDK for .NET. Notez, cependant, que ces valeurs sont différentes de celles utilisées par l'action du `Put Object` Amazon S3. Les Tools for Windows PowerShell prennent en charge les listes ACL conservées suivantes :

- NoACL
- privé
- public-read
- public-read-write
- aws-exec-read
- authenticated-read
- bucket-owner-read
- bucket-owner-full-control
- log-delivery-write

Pour plus d'informations sur ces paramètres de listes ACL prêtes à l'emploi, consultez Présentation de la liste de contrôle d'accès (ACL).

## Remarque concernant le chargement en plusieurs parties

Si vous utilisez l'API Amazon S3 pour charger un fichier qui dépasse 5 Go, vous devez utiliser le chargement en plusieurs parties. Cependant, l'applet de commande `Write-S3Object` fournie par les Tools for Windows PowerShell peut gérer de manière transparente les chargements de fichiers supérieurs à 5 Go.

Test du site Web

À ce stade, vous pouvez tester le site Web en y accédant à l'aide d'un navigateur. Les URL de sites Web statiques hébergés dans Amazon S3 suivent un format standard.

```
http://<bucket-name>.s3-website-<region>.amazonaws.com
```

Par exemple :

```
http://website-example.s3-website-us-west-1.amazonaws.com
```

Voir aussi

- Travaillez avec AWS les services du AWS Tools for PowerShell
- PutObject (Référence d'API Amazon S)
- Liste ACL prête à l'emploi (Référence d'API Amazon S)

## Suppression d'objets et de compartiments Amazon S3

Cette section décrit comment supprimer le site Web que vous avez créé dans les sections précédentes. Vous pouvez supprimer simplement les objets des fichiers HTML, puis supprimer le compartiment Amazon S3 du site.

Exécutez d'abord l'applet de commande `Remove-S3Object` pour supprimer les objets des fichiers HTML du compartiment Amazon S3.

```
PS > foreach ( $obj in "index.html", "error.html" ) {
>> Remove-S3Object -BucketName website-example -Key $obj
>> }
>>
IsDeleteMarker
--------------
False
```

La réponse `False` est un artefact attendu de la façon dont Amazon S3 traite la demande. Dans ce contexte, elle n'indique pas de problème.

Exécutez ensuite l'applet de commande `Remove-S3Bucket` pour supprimer le compartiment Amazon S3 désormais vide du site.

```
PS > Remove-S3Bucket -BucketName website-example

RequestId      : E480ED92A2EC703D
AmazonId2      : k6tqaqC1nMkoeYwbuJXUx1/UDa49BJd6dfLN0Ls1mWYNPHjbc8/Nyvm6AGbWcc2P
ResponseStream :
Headers        : {x-amz-id-2, x-amz-request-id, Date, Server}
Metadata       : {}
ResponseXml    :
```

Dans la version 1.1 et dans les versions plus récentes des AWS Tools for PowerShell, vous pouvez ajouter le paramètre `-DeleteBucketContent` à `Remove-S3Bucket`, ce qui supprime d'abord tous

les objets et versions d'objets du compartiment spécifié avant de tenter de supprimer le compartiment lui-même. Selon le nombre d'objets ou versions d'objet du compartiment, cette opération peut prendre un certain temps. Dans les versions des Tools for Windows PowerShell antérieures à la version 1.1, le compartiment devait être vide pour que `Remove-S3Bucket` puisse le supprimer.

> **Note**
>
> `-Force` vous invite à confirmer avant l'exécution de l'applet de commande, sauf si vous ajoutez le paramètre AWS Tools for PowerShell.

### Voir aussi

- Travaillez avec AWS les services du AWS Tools for PowerShell
- Delete Object (Référence d'API Amazon S)
- DeleteBucket (Référence d'API Amazon S)

## Charger du contenu du texte en ligne sur Amazon S3

L'applet de commande `Write-S3Object` permet de charger du contenu de texte en ligne dans Amazon S3. Avec le paramètre `-Content` (alias `-Text`), vous pouvez spécifier le contenu texte qui doit être chargé dans Amazon S3, sans avoir besoin de le placer d'abord dans un fichier. Le paramètre accepte les chaînes d'une ligne simples, ainsi que, comme ici, les chaînes contenant plusieurs lignes.

```
PS > # Specifying content in-line, single line text:
PS > write-s3object amzn-s3-demo-bucket -key myobject.txt -content "file content"

PS > # Specifying content in-line, multi-line text: (note final newline needed to end
 in-line here-string)
PS > write-s3object amzn-s3-demo-bucket -key myobject.txt -content @"
>> line 1
>> line 2
>> line 3
>> "@
>>
PS > # Specifying content from a variable: (note final newline needed to end in-line
 here-string)
PS > $x = @"
```

```
>> line 1
>> line 2
>> line 3
>> "@
>>
PS > write-s3object amzn-s3-demo-bucket -key myobject.txt -content $x
```

# Amazon EC2 et Tools for Windows PowerShell

Vous pouvez exécuter les tâches courantes relatives à Amazon EC2 à l'aide de AWS Tools for PowerShell.

Les exemples de commande présentés ici supposent que vous avez défini les informations d'identification par défaut et une région par défaut pour votre session PowerShell. Par conséquent, nous n'incluons pas d'informations d'identification ou de région lorsque nous appelons les applets de commande. Pour de plus amples informations, veuillez consulter . Premiers pas avec AWS Tools for Windows PowerShell.

Rubriques

- Création d'une paire de clés
- Création d'un groupe de sécurité à l'aide de Windows PowerShell
- Recherche d'une Amazon Machine Image avec Windows PowerShell
- Lancer une EC2 instance Amazon à l'aide de Windows PowerShell

## Création d'une paire de clés

L'exemple `New-EC2KeyPair` suivant crée une paire de clés et la stocke dans la variable PowerShell `$myPSKeyPair`

```
PS > $myPSKeyPair = New-EC2KeyPair -KeyName myPSKeyPair
```

Associez l'objet paire de clés à l'applet de commande `Get-Member` pour afficher la structure de l'objet.

```
PS > $myPSKeyPair | Get-Member

     TypeName: Amazon.EC2.Model.KeyPair
```

```
  Name                MemberType   Definition
  ----                ----------   ----------
  Equals              Method       bool Equals(System.Object obj)
  GetHashCode         Method       int GetHashCode()
  GetType             Method       type GetType()
  ToString            Method       string ToString()
  KeyFingerprint      Property     System.String KeyFingerprint {get;set;}
  KeyMaterial         Property     System.String KeyMaterial {get;set;}
  KeyName             Property     System.String KeyName {get;set;}
```

Associez l'objet paire de clés à l'applet de commande `Format-List` pour afficher les valeurs des membres `KeyName`, `KeyFingerprint` et `KeyMaterial`. (La sortie a été tronquée pour des raisons de lisibilité.)

```
PS > $myPSKeyPair | Format-List KeyName, KeyFingerprint, KeyMaterial

  KeyName        : myPSKeyPair
  KeyFingerprint : 09:06:70:8e:26:b6:e7:ef:8f:fe:4a:1d:bc:9c:6a:63:11:ac:ad:3c
  KeyMaterial    : ----BEGIN RSA PRIVATE KEY----
                   MIIEogIBAAKCAQEAkK+ANYUS9c7niNjYfaCn6KYj/D0I6djnFoQE...
                   Mz6btoxPcE7EMeH1wySUp8nouAS9xbl9l7+VkD74bN9KmNcPa/Mu...
                   Zyn4vVe0Q5il/MpkrRogHqOB0rigeTeV5Yc3lvO0RFFPu0Kz4kcm...
                   w3Jg8dKsWn0plOpX7V3sRC02KgJIbejQUvBFGi5OQK9bm4tXBIeC...
                   daxKIAQMtDUdmBDrhR1/YMv8itFe5DiLLbq7Ga+FDcS85NstBa3h...
                   iuskGkcvgWkcFQkLmRHRoDpPb+OdFsZtjHZDpMVFmA9tT8EdbkEF...
                   3SrNeqZPsxJJIxOodb3CxLJpg75JU5kyWnb0+sDNVHoJiZCULCr0...
                   GGlLfEgB95KjGIk7zEv2Q7K6s+DHclrDeMZWa7KFNRZuCuX7jssC...
                   xO98abxMr3o3TNU6p1ZYRJEQ0oJr0W+kc+/8SWb8NIwfLtwhmJEy...
                   1BX9X8WFX/A8VLHrT1elrKmLkNECgYEAwltkV1pOJAFhz9p7ZFEv...
                   vvVsPaF0Ev9bk9pqhx269PB5Ox2KokwCagDMMaYvasWobuLmNu/1...
                   lmwRx7KTeQ7W1J3OLgxHA1QNMkip9c4Tb3q9vVc3t/fPf8vwfJ8C...
                   63g6N6rk2FkHZX1E62BgbewUd3eZOS05Ip4VUdvtGcuc8/qa+e5C...
                   KXgyt9nl64pMv+VaXfXkZhdLAdY0Khc9TGB9++VMSG5TrD15YJId...
                   gYALEI7m1jJKpHWAEs0hiemw5VmKyIZpzGstSJsFStERlAjiETDH...
                   YAtnI4J8dRyP9I7BOVOn3wNfIjk85gi1/0Oc+j8S65giLAfndWGR...
                   9R9wIkm5BMUcSRRcDy0yuwKBgEbkOnGGSD0ah4HkvrUkepIbUDTD...
                   AnEBM1cXI5UT7BfKInpUihZi59QhgdK/hkOSmWhlZGWikJ5VizBf...
                   drkBr/vTKVRMTi3lVFB7KkIV1xJxC5E/BZ+YdZEpWoCZAoGAC/Cd...
                   TTld5N6opgOXAcQJwzqoGa9ZMwc5Q9f4bfRc67emkw0ZAAwSsvWR...
                   x3O2duuy7/smTwWwskEWRK5IrUxoMv/VVYaqdzcOajwieNrblr7c...
                   -----END RSA PRIVATE KEY-----
```

Le membre `KeyMaterial` stocke la clé privée de la paire de clés. La clé publique est stockée dans AWS. Vous ne pouvez pas récupérer la clé publique dans AWS, mais vous pouvez vérifier la clé publique en comparant la `KeyFingerprint` de la clé privée à celle renvoyée par AWS pour la clé publique.

### Affichage de l'empreinte de votre paire de clés

Vous pouvez utiliser l'applet de commande `Get-EC2KeyPair` pour afficher l'empreinte de votre paire de clés.

```
PS > Get-EC2KeyPair -KeyName myPSKeyPair | format-list KeyName, KeyFingerprint

  KeyName        : myPSKeyPair
  KeyFingerprint : 09:06:70:8e:26:b6:e7:ef:8f:fe:4a:1d:bc:9c:6a:63:11:ac:ad:3c
```

### Stockage de votre clé privée

Pour stocker la clé privée dans un fichier, associez le membre `KeyFingerMaterial` à l'applet de commande `Out-File`.

```
PS > $myPSKeyPair.KeyMaterial | Out-File -Encoding ascii myPSKeyPair.pem
```

Vous devez spécifier `-Encoding ascii` lors de l'écriture de la clé privée dans un fichier. Dans le cas contraire, des outils comme `openssl` peuvent ne pas être en mesure de lire le fichier correctement. Vous pouvez vérifier que le format du fichier obtenu est correct en utilisant une commande telle que la commande suivante :

```
PS > openssl rsa -check < myPSKeyPair.pem
```

(L'outil `openssl` n'est pas inclus dans les AWS Tools for PowerShell ni dans le AWS SDK for .NET.)

### Suppression de votre paire de clés

Vous avez besoin de votre paire de clés pour lancer une instance et vous y connecter. Lorsque vous aurez terminé d'utiliser une paire de clés, vous pourrez la supprimer. Pour supprimer la clé publique de AWS, utilisez l'applet de commande `Remove-EC2KeyPair`. Lorsque vous y serez invité, appuyez sur `Enter` pour supprimer la paire de clés.

```
PS > Remove-EC2KeyPair -KeyName myPSKeyPair
```

```
Confirm
Performing the operation "Remove-EC2KeyPair (DeleteKeyPair)" on target "myPSKeyPair".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "Y"):
```

La variable, `$myPSKeyPair`, existe toujours dans la session actuelle de PowerShell et contient toujours les informations sur la paire de clés. Le fichier `myPSKeyPair.pem` existe également. Cependant, la clé privée n'est plus valide, car la clé publique de la paire de clés n'est plus stockée dans AWS.

## Création d'un groupe de sécurité à l'aide de Windows PowerShell

Vous pouvez utiliser le AWS Tools for PowerShell pour créer et configurer un groupe de sécurité. La réponse est l'ID du groupe de sécurité.

Si vous devez vous connecter à votre instance, vous devez configurer le groupe de sécurité pour autoriser le SSH trafic (Linux) ou RDP le trafic (Windows).

Rubriques



### Prérequis

Vous avez besoin de l'adresse IP publique de votre ordinateur, CIDR notée. Vous pouvez obtenir l'adresse IP publique de votre ordinateur local à l'aide d'un service. Par exemple, Amazon fournit le service suivant : http://checkip.amazonaws.com/ ou https://checkip.amazonaws.com/. Pour trouver un autre service qui fournit votre adresse IP, utilisez l'expression de recherche « what is my IP address » (quelle est mon adresse IP). Si vous vous connectez via ISP ou derrière votre pare-feu sans adresse IP statique, vous devez rechercher la plage d'adresses IP pouvant être utilisées par vos ordinateurs clients.

> **Warning**
>
> Si vous spécifiez `0.0.0.0/0`, vous autorisez le trafic à partir de n'importe quelle adresse IP dans le monde. En ce qui concerne les RDP protocoles SSH et, vous pouvez considérer cela comme acceptable pendant une courte période dans un environnement de test, mais cela n'est pas sûr pour les environnements de production. En production, assurez-vous d'autoriser l'accès uniquement à partir de l'adresse IP individuelle ou de la plage d'adresses appropriée.

## Création d'un groupe de sécurité pour EC2 - VPC

> **Warning**
>
> EC2-Classic a été retiré le 15 août 2022. Nous vous recommandons de migrer de EC2 - Classic vers unVPC. Pour plus d'informations, consultez le billet de blog intitulé « EC2Classic Networking is Retiring — Here's How to Prepare ».

L'`New-EC2SecurityGroup`exemple suivant ajoute le `-VpcId` paramètre pour créer un groupe de sécurité pour le groupe spécifiéVPC.

```
PS > $groupid = New-EC2SecurityGroup `
    -VpcId "vpc-da0013b3" `
    -GroupName "myPSSecurityGroup" `
    -GroupDescription "EC2-VPC from PowerShell"
```

Pour afficher la configuration initiale du groupe de sécurité, utilisez l'applet de commande `Get-EC2SecurityGroup`. Par défaut, le groupe de sécurité d'un VPC contient une règle qui autorise tout le trafic sortant. Notez que vous ne pouvez pas référencer un groupe de sécurité pour EC2 - VPC par son nom.

```
PS > Get-EC2SecurityGroup -GroupId sg-5d293231

OwnerId             : 123456789012
GroupName           : myPSSecurityGroup
GroupId             : sg-5d293231
Description         : EC2-VPC from PowerShell
IpPermissions       : {}
IpPermissionsEgress : {Amazon.EC2.Model.IpPermission}
VpcId               : vpc-da0013b3
Tags                : {}
```

Pour définir les autorisations pour le trafic entrant sur le TCP port 22 (SSH) et le TCP port 3389, utilisez l'`New-Object`applet de commande. L'exemple de script suivant définit les autorisations pour les TCP ports 22 et 3389 à partir d'une seule adresse IP,`203.0.113.25/32`.

```
$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 $groupid -IpPermissions @( $ip1, $ip2 )
```

Pour vérifier que le groupe de sécurité a été mis à jour, utilisez à nouveau l'applet de commande `Get-EC2SecurityGroup`.

```
PS > Get-EC2SecurityGroup -GroupIds sg-5d293231

OwnerId             : 123456789012
GroupName           : myPSSecurityGroup
GroupId             : sg-5d293231
Description         : EC2-VPC from PowerShell
IpPermissions       : {Amazon.EC2.Model.IpPermission}
IpPermissionsEgress : {Amazon.EC2.Model.IpPermission}
VpcId               : vpc-da0013b3
Tags                : {}
```

Pour afficher les règles de trafic entrant, vous pouvez récupérer la propriété `IpPermissions` de l'objet de collection renvoyé par la commande précédente.

```
PS > (Get-EC2SecurityGroup -GroupIds sg-5d293231).IpPermissions

IpProtocol       : tcp
FromPort         : 22
ToPort           : 22
UserIdGroupPairs : {}
IpRanges         : {203.0.113.25/32}

IpProtocol       : tcp
FromPort         : 3389
ToPort           : 3389
UserIdGroupPairs : {}
IpRanges         : {203.0.113.25/32}
```

# Recherche d'une Amazon Machine Image avec Windows PowerShell

Lorsque vous lancez une instance Amazon EC2, vous spécifiez une Amazon Machine Image (AMI) qui sert de modèle à l'instance. Cependant, les ID des AMI AWS Windows changent fréquemment, car AWS fournit de nouvelles AMI avec les dernières mises à jour et les améliorations de sécurité. Vous pouvez utiliser les applets de commande Get-EC2Image et Get-EC2ImageByName pour rechercher les AMI Windows actuelles et obtenir leurs ID.

Rubriques



## Get-EC2Image

L'applet de commande `Get-EC2Image` récupère une liste d'AMI que vous pouvez utiliser.

Utilisez le paramètre `-Owner` avec la valeur de tableau `amazon, self` pour que `Get-EC2Image` récupère uniquement les AMI qui appartiennent à Amazon ou à vous-même. Dans ce contexte, vous faites référence à l'utilisateur dont vous avez utilisé les informations d'identification pour appeler l'applet de commande.

```
PS > Get-EC2Image -Owner amazon, self
```

Vous pouvez délimiter les résultats à l'aide du paramètre `-Filter`. Pour spécifier le filtre, créez un objet de type `Amazon.EC2.Model.Filter`. Par exemple, utilisez le filtre suivant pour afficher uniquement les AMI Windows.

```
$platform_values = New-Object 'collections.generic.list[string]'
$platform_values.add("windows")
$filter_platform = New-Object Amazon.EC2.Model.Filter -Property @{Name = "platform";
 Values = $platform_values}
Get-EC2Image -Owner amazon, self -Filter $filter_platform
```

Voici un exemple de l'une des AMI renvoyées par l'applet de commande ; la sortie réelle de la commande précédente fournit des informations pour de nombreuses AMI.

```
Architecture        : x86_64
BlockDeviceMappings : {/dev/sda1, xvdca, xvdcb, xvdcc…}
```

```
CreationDate        : 2019-06-12T10:41:31.000Z
Description         : Microsoft Windows Server 2019 Full Locale English with SQL Web
 2017 AMI provided by Amazon
EnaSupport          : True
Hypervisor          : xen
ImageId             : ami-000226b77608d973b
ImageLocation       : amazon/Windows_Server-2019-English-Full-SQL_2017_Web-2019.06.12
ImageOwnerAlias     : amazon
ImageType           : machine
KernelId            :
Name                : Windows_Server-2019-English-Full-SQL_2017_Web-2019.06.12
OwnerId             : 801119661308
Platform            : Windows
ProductCodes        : {}
Public              : True
RamdiskId           :
RootDeviceName      : /dev/sda1
RootDeviceType      : ebs
SriovNetSupport     : simple
State               : available
StateReason         :
Tags                : {}
VirtualizationType  : hvm
```

## Get-EC2ImageByName

L'applet de commande `Get-EC2ImageByName` vous permet de filtrer la liste des AMI Windows AWS en fonction du type de configuration serveur qui vous intéresse.

Lorsqu'elle est exécutée sans paramètres, comme suit, l'applet de commande émet l'ensemble complet des noms de filtre actuels :

```
PS > Get-EC2ImageByName

WINDOWS_2016_BASE
WINDOWS_2016_NANO
WINDOWS_2016_CORE
WINDOWS_2016_CONTAINER
WINDOWS_2016_SQL_SERVER_ENTERPRISE_2016
WINDOWS_2016_SQL_SERVER_STANDARD_2016
WINDOWS_2016_SQL_SERVER_WEB_2016
WINDOWS_2016_SQL_SERVER_EXPRESS_2016
WINDOWS_2012R2_BASE
```

```
WINDOWS_2012R2_CORE
WINDOWS_2012R2_SQL_SERVER_EXPRESS_2016
WINDOWS_2012R2_SQL_SERVER_STANDARD_2016
WINDOWS_2012R2_SQL_SERVER_WEB_2016
WINDOWS_2012R2_SQL_SERVER_EXPRESS_2014
WINDOWS_2012R2_SQL_SERVER_STANDARD_2014
WINDOWS_2012R2_SQL_SERVER_WEB_2014
WINDOWS_2012_BASE
WINDOWS_2012_SQL_SERVER_EXPRESS_2014
WINDOWS_2012_SQL_SERVER_STANDARD_2014
WINDOWS_2012_SQL_SERVER_WEB_2014
WINDOWS_2012_SQL_SERVER_EXPRESS_2012
WINDOWS_2012_SQL_SERVER_STANDARD_2012
WINDOWS_2012_SQL_SERVER_WEB_2012
WINDOWS_2012_SQL_SERVER_EXPRESS_2008
WINDOWS_2012_SQL_SERVER_STANDARD_2008
WINDOWS_2012_SQL_SERVER_WEB_2008
WINDOWS_2008R2_BASE
WINDOWS_2008R2_SQL_SERVER_EXPRESS_2012
WINDOWS_2008R2_SQL_SERVER_STANDARD_2012
WINDOWS_2008R2_SQL_SERVER_WEB_2012
WINDOWS_2008R2_SQL_SERVER_EXPRESS_2008
WINDOWS_2008R2_SQL_SERVER_STANDARD_2008
WINDOWS_2008R2_SQL_SERVER_WEB_2008
WINDOWS_2008RTM_BASE
WINDOWS_2008RTM_SQL_SERVER_EXPRESS_2008
WINDOWS_2008RTM_SQL_SERVER_STANDARD_2008
WINDOWS_2008_BEANSTALK_IIS75
WINDOWS_2012_BEANSTALK_IIS8
VPC_NAT
```

Pour restreindre l'ensemble des images renvoyées, spécifiez un ou plusieurs noms de filtre à l'aide du paramètre `Names`.

```
PS > Get-EC2ImageByName -Names WINDOWS_2016_CORE

Architecture        : x86_64
BlockDeviceMappings : {/dev/sda1, xvdca, xvdcb, xvdcc…}
CreationDate        : 2019-08-16T09:36:09.000Z
Description         : Microsoft Windows Server 2016 Core Locale English AMI provided by
 Amazon
EnaSupport          : True
Hypervisor          : xen
```

```
ImageId             : ami-06f2a2afca06f15fc
ImageLocation       : amazon/Windows_Server-2016-English-Core-Base-2019.08.16
ImageOwnerAlias     : amazon
ImageType           : machine
KernelId            :
Name                : Windows_Server-2016-English-Core-Base-2019.08.16
OwnerId             : 801119661308
Platform            : Windows
ProductCodes        : {}
Public              : True
RamdiskId           :
RootDeviceName      : /dev/sda1
RootDeviceType      : ebs
SriovNetSupport     : simple
State               : available
StateReason         :
Tags                : {}
VirtualizationType  : hvm
```

## Lancer une EC2 instance Amazon à l'aide de Windows PowerShell

Pour lancer une EC2 instance Amazon, vous avez besoin de la paire de clés et du groupe de sécurité que vous avez créés dans les sections précédentes. Vous avez également besoin de l'ID d'une Amazon Machine Image (AMI). Pour plus d'informations, consultez la documentation de suivante :

- Création d'une paire de clés
- Création d'un groupe de sécurité à l'aide de Windows PowerShell
- Rechercher une image de machine Amazon à l'aide de Windows PowerShell

> **Important**
>
> Si vous lancez une instance non comprise dans l'offre gratuite, vous serez facturé dès le lancement de l'instance et pendant toute la durée d'exécution de l'instance, même si elle demeure inactive.

### Rubriques

- Lancement d'une instance dans un VPC
- Lancement d'une instance Spot dans un VPC

## Lancement d'une instance dans un VPC

> **Warning**
>
> EC2-Classic a été retiré le 15 août 2022. Nous vous recommandons de migrer de EC2 - Classic vers unVPC. Pour plus d'informations, consultez le billet de blog intitulé « EC2Classic Networking is Retiring — Here's How to Prepare ».

La commande suivante crée une seule instance `m1.small` dans le sous-réseau privé spécifié. Le groupe de sécurité doit être valide pour le sous-réseau spécifié.

```
PS > New-EC2Instance `
    -ImageId ami-c49c0dac `
    -MinCount 1 -MaxCount 1 `
    -KeyName myPSKeyPair `
    -SecurityGroupId sg-5d293231 `
    -InstanceType m1.small `
    -SubnetId subnet-d60013bf

ReservationId   : r-b70a0ef1
OwnerId         : 123456789012
RequesterId     :
Groups          : {}
GroupName       : {}
Instances       : {}
```

Au départ, votre instance est à l'état `pending`, mais elle passe à l'état `running` en quelques minutes. Pour afficher les informations sur votre instance, utilisez l'applet de commande `Get-EC2Instance`. Si vous avez plusieurs instances, vous pouvez filtrer les résultats sur l'ID de réservation à l'aide du paramètre `Filter`. Tout d'abord, créez un objet de type `Amazon.EC2.Model.Filter`. Ensuite, appelez `Get-EC2Instance` qui utilise le filtre, puis affiche la propriété `Instances`.

```
PS > $reservation = New-Object 'collections.generic.list[string]'
PS > $reservation.add("r-b70a0ef1")
PS > $filter_reservation = New-Object Amazon.EC2.Model.Filter -Property @{Name =
 "reservation-id"; Values = $reservation}
PS > (Get-EC2Instance -Filter $filter_reservation).Instances

AmiLaunchIndex        : 0
```

```
Architecture          : x86_64
BlockDeviceMappings   : {/dev/sda1}
ClientToken           :
EbsOptimized          : False
Hypervisor            : xen
IamInstanceProfile    :
ImageId               : ami-c49c0dac
InstanceId            : i-5203422c
InstanceLifecycle     :
InstanceType          : m1.small
KernelId              :
KeyName               : myPSKeyPair
LaunchTime            : 12/2/2018 3:38:52 PM
Monitoring            : Amazon.EC2.Model.Monitoring
NetworkInterfaces     : {}
Placement             : Amazon.EC2.Model.Placement
Platform              : Windows
PrivateDnsName        :
PrivateIpAddress      : 10.25.1.11
ProductCodes          : {}
PublicDnsName         :
PublicIpAddress       : 198.51.100.245
RamdiskId             :
RootDeviceName        : /dev/sda1
RootDeviceType        : ebs
SecurityGroups        : {myPSSecurityGroup}
SourceDestCheck       : True
SpotInstanceRequestId :
SriovNetSupport       :
State                 : Amazon.EC2.Model.InstanceState
StateReason           :
StateTransitionReason :
SubnetId              : subnet-d60013bf
Tags                  : {}
VirtualizationType    : hvm
VpcId                 : vpc-a01106c2
```

## Lancement d'une instance Spot dans un VPC

L'exemple de script suivant demande une instance Spot dans le sous-réseau spécifié. Le groupe de sécurité doit être celui que vous avez créé pour le et VPC qui contient le sous-réseau spécifié.

```
$interface1 = New-Object Amazon.EC2.Model.InstanceNetworkInterfaceSpecification
```

```
$interface1.DeviceIndex = 0
$interface1.SubnetId = "subnet-b61f49f0"
$interface1.PrivateIpAddress = "10.0.1.5"
$interface1.Groups.Add("sg-5d293231")
Request-EC2SpotInstance `
    -SpotPrice 0.007 `
    -InstanceCount 1 `
    -Type one-time `
    -LaunchSpecification_ImageId ami-7527031c `
    -LaunchSpecification_InstanceType m1.small `
    -Region us-west-2 `
    -LaunchSpecification_NetworkInterfaces $interface1
```

# AWS Lambda et AWS Tools for PowerShell

Le module AWSLambdaPSCore vous permet de développer des fonctions AWS Lambda dans PowerShell Core 6.0 à l'aide du runtime .NET Core 2.1. Les développeurs PowerShell peuvent gérer des ressources AWSet écrire des scripts d'automatisation dans l'environnement PowerShell en utilisant Lambda. La prise en charge de PowerShell dans Lambda vous permet d'exécuter des fonctions ou des scripts PowerShell en réponse à tout événement Lambda, tel qu'un événement Amazon S3 ou un événement Amazon CloudWatch planifié. Le module AWSLambdaPSCore est un module AWS distinct pour PowerShell ; il ne fait pas partie de AWS Tools for PowerShell, l'installation du module AWSLambdaPSCore ne provoque pas non plus l'installation de AWS Tools for PowerShell.

Après avoir installé le module AWSLambdaPSCore, vous pourrez utiliser toutes les applets de commande PowerShell disponibles (ou développer les vôtres) pour créer des fonctions sans serveur. Le module AWS Lambda Tools for PowerShell inclut des modèles de projet pour les applications sans serveur reposant sur PowerShell, ainsi que des outils pour publier des projets sur AWS.

Le support du module AWSLambdaPSCore est disponible dans toutes les régions qui prennent en charge Lambda. Pour en savoir plus sur les régions prises en charge, consultez le AWSTableau des régions.

## Prérequis

Les étapes suivantes sont obligatoires pour pouvoir installer et utiliser le module AWSLambdaPSCore. Pour plus de détails sur ces étapes, consultez la section relative à la configuration d'un environnement de développement PowerShell dans le guide du développeur AWS Lambda.

- Installez la bonne version de PowerShell : la prise en charge de PowerShell dans Lambda repose sur la version inter-plateforme PowerShell Core 6.0. Vous pouvez développer des fonctions PowerShell Lambda sous Windows, Linux ou Mac. Si vous ne disposez pas d'au moins cette version de PowerShell, vous trouverez les instructions sur le site Web de la documentation Microsoft PowerShell.
- Installer le kit SDK .NET Core 2.1 : étant donné que PowerShell Core repose sur .NET Core, la prise en charge de PowerShell dans Lambda utilise le même runtime .NET Core 2.1 Lambda pour les fonctions .NET Core et Lambda PowerShell. Les applets de commande de publication Lambda PowerShell utilisent le kit SDK .NET Core 2.1 pour créer le package de déploiement Lambda. Le kit SDK .NET Core 2.1 est disponible à partir du Centre de téléchargement Microsoft. Veillez à installer le kit SDK, et non le runtime.

### Installation du module AWSLambdaPSCore

Une fois les conditions requises respectées, vous êtes prêt à installer le module AWSLambdaPSCore. Exécutez la commande suivante dans une session PowerShell Core.

```
PS> Install-Module AWSLambdaPSCore -Scope CurrentUser
```

Vous êtes prêt à commencer à développer des fonctions Lambda dans PowerShell. Pour plus d'informations sur la mise en route, consultez la section relative au modèle de programmation pour la création de fonctions Lambda dans PowerShell dans le guide du développeur AWS Lambda.

### Voir aussi

- Annonce de la prise en charge de PowerShell Core par Lambda sur le blog des développeurs AWS
- Module AWSLambdaPSCore sur le site web PowerShell Gallery
- Configuration d'un environnement de développement PowerShell
- AWS Lambda Tools for Powershell sur GitHub
- Console Lambda AWS

## Amazon SQS, Amazon SNS et Tools for Windows PowerShell

Cette section fournit des exemples qui montrent comment :

- Créer une file d'attente Amazon SQS et obtenir l'ARN (Amazon Resource Name) de la file d'attente.
- Créez une rubrique Amazon SNS.
- Accordez les autorisations à la rubrique SNS afin qu'elle puisse envoyer des messages à la file d'attente.
- Abonner la file d'attente à la rubrique SNS
- Accordez aux utilisateurs IAM ou aux comptes AWS les autorisations d'effectuer une publication dans la rubrique SNS et de lire les messages à partir de la file d'attente SQS.
- Vérifiez les résultats en publiant un message dans la rubrique et en lisant le message à partir de la file d'attente.

## Créer une file d'attente Amazon SQS et obtenir l'ARN de la file d'attente

La commande suivante crée une file d'attente SQS dans votre région par défaut. La sortie affiche l'URL de la nouvelle file d'attente.

```
PS > New-SQSQueue -QueueName myQueue
https://sqs.us-west-2.amazonaws.com/123456789012/myQueue
```

La commande suivante récupère l'ARN de la file d'attente.

```
PS > Get-SQSQueueAttribute -QueueUrl https://sqs.us-west-2.amazonaws.com/123456789012/
myQueue -AttributeName QueueArn
...
QueueARN              : arn:aws:sqs:us-west-2:123456789012:myQueue
...
```

## Créer une rubrique Amazon SNS

La commande suivante crée une rubrique SNS dans votre région par défaut et renvoie l'ARN de la nouvelle rubrique.

```
PS > New-SNSTopic -Name myTopic
arn:aws:sns:us-west-2:123456789012:myTopic
```

## Accorder les autorisations à la rubrique SNS

L'exemple de script suivant crée à la fois une file d'attente SQS et une rubrique SNS, et accorde des autorisations à la rubrique SNS afin qu'elle puisse envoyer des messages à la file d'attente SQS :

```
# create the queue and topic to be associated
$qurl = New-SQSQueue -QueueName "myQueue"
$topicarn = New-SNSTopic -Name "myTopic"

# get the queue ARN to inject into the policy; it will be returned
# in the output's QueueARN member but we need to put it into a variable
# so text expansion in the policy string takes effect
$qarn = (Get-SQSQueueAttribute -QueueUrl $qurl -AttributeNames "QueueArn").QueueARN

# construct the policy and inject arns
$policy = @"
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Principal": "*",
        "Action": "SQS:SendMessage",
        "Resource": "$qarn",
        "Condition": { "ArnEquals": { "aws:SourceArn": "$topicarn" } }
    }
}
"@

# set the policy
Set-SQSQueueAttribute -QueueUrl $qurl -Attribute @{ Policy=$policy }
```

## Abonner la file d'attente à la rubrique SNS

La commande suivante abonne la file d'attente `myQueue` à la rubrique SNS `myTopic` et renvoie l'ID d'abonnement :

```
PS > Connect-SNSNotification `
    -TopicARN arn:aws:sns:us-west-2:123456789012:myTopic `
    -Protocol SQS `
    -Endpoint arn:aws:sqs:us-west-2:123456789012:myQueue
arn:aws:sns:us-west-2:123456789012:myTopic:f8ff77c6-e719-4d70-8e5c-a54d41feb754
```

## Accorder les autorisations

La commande suivante donne l'autorisation d'effectuer l'action `sns:Publish` sur la rubrique `myTopic` :

```
PS > Add-SNSPermission `
    -TopicArn arn:aws:sns:us-west-2:123456789012:myTopic `
    -Label ps-cmdlet-topic `
    -AWSAccountIds 123456789012 `
    -ActionNames publish
```

La commande suivante donne l'autorisation d'effectuer les actions `sqs:ReceiveMessage` et `sqs:DeleteMessage` sur la file d'attente `myQueue`.

```
PS > Add-SQSPermission `
    -QueueUrl https://sqs.us-west-2.amazonaws.com/123456789012/myQueue `
    -AWSAccountId "123456789012" `
    -Label queue-permission `
    -ActionName SendMessage, ReceiveMessage
```

## Vérifier les résultats

La commande suivante teste votre nouvelle file d'attente et votre rubrique en publiant un message dans la rubrique SNS `myTopic`, puis renvoie le `MessageId`.

```
PS > Publish-SNSMessage `
    -TopicArn arn:aws:sns:us-west-2:123456789012:myTopic `
    -Message "Have A Nice Day!"
728180b6-f62b-49d5-b4d3-3824bb2e77f4
```

La commande suivante récupère le message dans la file d'attente SQS `myQueue` et l'affiche.

```
PS > Receive-SQSMessage -QueueUrl https://sqs.us-west-2.amazonaws.com/123456789012/
myQueue

Attributes             : {}
Body                   : {
                           "Type" : "Notification",
                           "MessageId" : "491c687d-b78d-5c48-b7a0-3d8d769ee91b",
                           "TopicArn" : "arn:aws:sns:us-west-2:123456789012:myTopic",
                           "Message" : "Have A Nice Day!",
```

```
                            "Timestamp" : "2019-09-09T21:06:27.201Z",
                            "SignatureVersion" : "1",
                            "Signature" :
 "llE17A2+XOuJZnw3TlgcXz4C4KPLXZxbxoEMIirelhl3u/oxkWmz5+9tJKFMns1ZOqQvKxk
+ExfEZcD5yWt6biVuBb8pyRmZ1bO3hUENl3ayv2WQiQT1vpLpM7VEQN5m+hLIiPFcs
                           vyuGkJReV7lOJWPHnCN
+qTE2lId2RPkFOeGtLGawTsSPTWEvJdDbLlf7E0zZ0q1niXTUtpsZ8Swx01X3QO6u9i9qBFt0ekJFZNJp6Avu05hIklb4yo
y0a8Yl9lWp7a7EoWaBn0zhCESe7o
                           kZC6ncBJWphX7KCGVYD0qhVf/5VDgBuv9w8T+higJyvr3WbaSvg==",
                            "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/
SimpleNotificationService-6aad65c2f9911b05cd53efda11f913f9.pem",
                            "UnsubscribeURL" :
                           "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-west-2:123456789012:myTopic:22b77de7-
a216-4000-9a23-bf465744ca84"
                           }
MD5OfBody              : 5b5ee4f073e9c618eda3718b594fa257
MD5OfMessageAttributes :
MessageAttributes      : {}
MessageId              : 728180b6-f62b-49d5-b4d3-3824bb2e77f4
ReceiptHandle          :
 AQEB2vvk1e5cOKFjeIWJticabkc664yuDEjhucnIOqdVUmie7bX7GiJbl7F0enABUgaI2XjEcNPxixhVc/
wfsAJZLNHnl8SlbQa0R/kD+Saqa4OIvfj8x3M4Oh1yM1cVKpYmhAzsYrAwAD5g5FvxNBD6zs
                           +HmXdkax2Wd+9AxrHlQZV5ur1MoByKWWbDbsqoYJTJquCclOgWIak/sBx/
daBRMTiVQ4GHsrQWMVHtNC14q7Jy/0L2dkmb4dzJfJq0VbFSX1G+u/lrSLpgae+Dfux646y8yFiPFzY4ua4mCF/
SVUn63Spy
                           sHN12776axknhg3j9K/Xwj54DixdsegnrKoLx+ctI
+0jzAetBR66Q1VhIoJAq7s0a2MseyOeM/Jjucg6Sr9VUnTWVhV8ErXmotoiEg==
```

# CloudWatch depuis le AWS Tools for Windows PowerShell

Cette section montre un exemple d'utilisation des Tools for Windows PowerShell pour publier des données de métriques personnalisées sur CloudWatch.

Cet exemple présume que vous avez défini les informations d'identification par défaut et une région par défaut pour votre session PowerShell.

## Publication d'une métrique personnalisée sur votre tableau de bord CloudWatch

Le code PowerShell suivant initialise un objet CloudWatch `MetricDatum` et le publie sur le service. Vous pouvez voir le résultat de cette opération en accédant à la console CloudWatch.

```
$dat = New-Object Amazon.CloudWatch.Model.MetricDatum
$dat.Timestamp = (Get-Date).ToUniversalTime()
$dat.MetricName = "New Posts"
$dat.Unit = "Count"
$dat.Value = ".50"
Write-CWMetricData -Namespace "Usage Metrics" -MetricData $dat
```

Remarques :

- Les informations de date et d'heure que vous utilisez pour initialiser `$dat.Timestamp` doivent être au format UTC.
- La valeur que vous utilisez pour initialiser `$dat.Value` peut être une valeur de chaîne entourée de guillemets ou une valeur numérique (sans guillemets). L'exemple illustre une valeur de chaîne.

### Voir aussi

- Travaillez avec AWS les services du AWS Tools for PowerShell
- AmazonCloudWatchClient.PutMetricData (Référence du kit SDK .NET)
- MetricDatum (Référence d'API de service)
- Console Amazon CloudWatch

## Utilisation du paramètre ClientConfig dans les applets de commande

Le paramètre `ClientConfig` permet de spécifier certains paramètres de configuration lorsque vous vous connectez à un service. La plupart des propriétés possibles de ce paramètre sont définies dans la classe `Amazon.Runtime.ClientConfig`, qui est héritée dans les API pour les services AWS. Pour un exemple d'héritage simple, consultez la classe `Amazon.Keyspaces.AmazonKeyspacesConfig`. En outre, certains services définissent des propriétés supplémentaires qui ne sont appropriées que pour ce service. Pour obtenir un exemple de propriétés définies supplémentaires, consultez la classe `Amazon.S3.AmazonS3Config`, plus précisément la propriété `ForcePathStyle`.

## Utilisation du paramètre **`ClientConfig`**

Pour utiliser le paramètre `ClientConfig`, vous pouvez le spécifier sur la ligne de commande en tant qu'objet `ClientConfig` ou utiliser le splatting PowerShell pour transmettre une collection de valeurs de paramètres à une commande en tant qu'unité. Ces méthodes sont présentées dans les exemples suivants. Les exemples supposent que le module `AWS.Tools.S3` a été installé et importé, et que vous disposez d'un profil d'informations d'identification `[default]` avec les autorisations appropriées.

Définition d'un objet **`ClientConfig`**

```
$s3Config = New-Object -TypeName Amazon.S3.AmazonS3Config
$s3Config.ForcePathStyle = $true
$s3Config.Timeout = [TimeSpan]::FromMilliseconds(150000)
Get-S3Object -BucketName <BUCKET_NAME> -ClientConfig $s3Config
```

Ajout de propriétés **`ClientConfig`** en utilisant le splatting PowerShell

```
$params=@{
    ClientConfig=@{
        ForcePathStyle=$true
        Timeout=[TimeSpan]::FromMilliseconds(150000)
    }
    BucketName="<BUCKET_NAME>"
}

Get-S3Object @params
```

## Utilisation d'une propriété non définie

Lors de l'utilisation du splatting PowerShell, si vous spécifiez une propriété `ClientConfig` qui n'existe pas, le paramètre AWS Tools for PowerShell ne détecte pas l'erreur avant l'exécution, et renvoie alors une exception. En modifiant l'exemple ci-dessus :

```
$params=@{
    ClientConfig=@{
        ForcePathStyle=$true
        UndefinedProperty="Value"
        Timeout=[TimeSpan]::FromMilliseconds(150000)
    }
```

```
    BucketName="<BUCKET_NAME>"
}

Get-S3Object @params
```

Cet exemple produit une exception semblable à la suivante :

```
Cannot bind parameter 'ClientConfig'. Cannot create object of type
 "Amazon.S3.AmazonS3Config". The UndefinedProperty property was not found for the
 Amazon.S3.AmazonS3Config object.
```

## Définition de la Région AWS

Vous pouvez utiliser le paramètre `ClientConfig` pour définir la Région AWS pour la commande. La région est définie à travers la propriété `RegionEndpoint`. Le paramètre AWS Tools for PowerShell calcule la région à utiliser selon la priorité suivante :

1. Paramètre `-Region`
2. Région transmise dans le paramètre `ClientConfig`
3. État de la session PowerShell
4. Fichier AWS `config` partagé
5. Variables d'environnement
6. Métadonnées de l'instance Amazon EC2, si elles sont activées.

# Outils pour des exemples PowerShell de code

Les exemples de code présentés dans cette rubrique vous montrent comment utiliser le AWS Tools for PowerShell with AWS.

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

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

Les scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un service ou en les combinant à d'autres Services AWS.

Services

- ACMexemples utilisant des outils pour PowerShell
- Exemples d'Application Auto Scaling utilisant des outils pour PowerShell
- AppStream Exemples 2.0 utilisant des outils pour PowerShell
- Exemples d'Aurora utilisant des outils pour PowerShell
- Exemples d'Auto Scaling utilisant des outils pour PowerShell
- AWS Budgets exemples utilisant des outils pour PowerShell
- AWS Cloud9 exemples utilisant des outils pour PowerShell
- AWS CloudFormation exemples utilisant des outils pour PowerShell
- CloudFront exemples utilisant des outils pour PowerShell
- CloudTrail exemples utilisant des outils pour PowerShell
- CloudWatch exemples utilisant des outils pour PowerShell
- CodeCommit exemples utilisant des outils pour PowerShell
- CodeDeploy exemples utilisant des outils pour PowerShell
- CodePipeline exemples utilisant des outils pour PowerShell
- Exemples d'Amazon Cognito Identity utilisant des outils pour PowerShell

- AWS Config exemples utilisant des outils pour PowerShell
- Exemples de Device Farm utilisant des outils pour PowerShell
- AWS Directory Service exemples utilisant des outils pour PowerShell
- AWS DMS exemples utilisant des outils pour PowerShell
- Exemples DynamoDB utilisant des outils pour PowerShell
- EC2Exemples Amazon utilisant des outils pour PowerShell
- ECRExemples Amazon utilisant des outils pour PowerShell
- ECSExemples Amazon utilisant des outils pour PowerShell
- EFSExemples Amazon utilisant des outils pour PowerShell
- EKSExemples Amazon utilisant des outils pour PowerShell
- Elastic Load Balancing - Exemples de version 1 utilisant des outils pour PowerShell
- Elastic Load Balancing - Exemples de version 2 utilisant des outils pour PowerShell
- FSxExemples Amazon utilisant des outils pour PowerShell
- AWS Glue exemples utilisant des outils pour PowerShell
- AWS Health exemples utilisant des outils pour PowerShell
- IAMexemples utilisant des outils pour PowerShell
- Exemples de Kinesis utilisant des outils pour PowerShell
- Exemples Lambda utilisant des outils pour PowerShell
- Exemples d'Amazon ML utilisant des outils pour PowerShell
- Exemples de Macie utilisant des outils pour PowerShell
- AWS OpsWorks exemples utilisant des outils pour PowerShell
- AWS Price List exemples utilisant des outils pour PowerShell
- Exemples de groupes de ressources utilisant des outils pour PowerShell
- Resource Groups : API exemples de balisage à l'aide de Tools for PowerShell
- Exemples de Route 53 utilisant des outils pour PowerShell
- Exemples d'Amazon S3 utilisant des outils pour PowerShell
- Exemples de S3 Glacier utilisant des outils pour PowerShell
- SESExemples Amazon utilisant des outils pour PowerShell
- SNSExemples Amazon utilisant des outils pour PowerShell

- SQSExemples Amazon utilisant des outils pour PowerShell
- AWS STS exemples utilisant des outils pour PowerShell
- AWS Support exemples utilisant des outils pour PowerShell
- Exemples de Systems Manager utilisant des outils pour PowerShell
- Exemples d'Amazon Translate utilisant des outils pour PowerShell
- AWS WAFV2 exemples utilisant des outils pour PowerShell
- WorkSpaces exemples utilisant des outils pour PowerShell

# ACMexemples utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS Tools for PowerShell withACM.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `Get-ACMCertificate`

L'exemple de code suivant montre comment utiliser`Get-ACMCertificate`.

Outils pour PowerShell

Exemple 1 : Cet exemple montre comment renvoyer un certificat et sa chaîne à l'aide ARN du certificat.

```
Get-ACMCertificate -CertificateArn "arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
```

- Pour API plus de détails, consultez la section GetCertificateRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ACMCertificateDetail`

L'exemple de code suivant montre comment utiliser`Get-ACMCertificateDetail`.

Outils pour PowerShell

Exemple 1 : renvoie les détails du certificat spécifié.

```
Get-ACMCertificateDetail -CertificateArn "arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
```

Sortie :

```
CertificateArn          : arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
CreatedAt               : 1/21/2016 5:55:59 PM
DomainName              : www.example.com
DomainValidationOptions : {www.example.com}
InUseBy                 : {}
IssuedAt                : 1/1/0001 12:00:00 AM
Issuer                  :
KeyAlgorithm            : RSA-2048
NotAfter                : 1/1/0001 12:00:00 AM
NotBefore               : 1/1/0001 12:00:00 AM
RevocationReason        :
RevokedAt               : 1/1/0001 12:00:00 AM
Serial                  :
SignatureAlgorithm      : SHA256WITHRSA
Status                  : PENDING_VALIDATION
Subject                 : CN=www.example.com
SubjectAlternativeNames : {www.example.net}
```

- Pour API plus de détails, consultez la section DescribeCertificateRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ACMCertificateList`

L'exemple de code suivant montre comment utiliser`Get-ACMCertificateList`.

Exemple 1 : récupère la liste de tous vos certificats ARNs et le nom de domaine de chacun d'entre eux. L'applet de commande pagine automatiquement pour récupérer tous les. ARNs Pour contrôler manuellement la pagination, utilisez le MaxItem paramètre - pour contrôler le nombre de certificats ARNs renvoyés pour chaque appel de service et le NextToken paramètre - pour indiquer le point de départ de chaque appel.

```
Get-ACMCertificateList
```

Sortie :

```
CertificateArn
 DomainName
--------------
 ----------
arn:aws:acm:us-east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
 www.example.com
```

Exemple 2 : récupère une liste de tous vos certificats ARNs dont le statut correspond aux états fournis.

```
Get-ACMCertificateList -CertificateStatus "VALIDATION_TIMED_OUT","FAILED"
```

Exemple 3 : Cet exemple renvoie une liste de tous les certificats de la région us-east-1 dont le type de clé est _2048 et dont l'utilisation ou l'objectif étendu RSA de la clé est _. CODE SIGNING Vous trouverez les valeurs de ces paramètres de filtrage dans la rubrique de API référence ListCertificates relative aux filtres : https://docs.aws.amazon.com/acm/ latest/APIReference/ API_Filters.html.

```
Get-ACMCertificateList -Region us-east-1 -Includes_KeyType RSA_2048 -
Includes_ExtendedKeyUsage CODE_SIGNING
```

Sortie :

```
CertificateArn
 DomainName
--------------
 ----------
```

```
arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-d7c0-48c1-af8d-2133d8f30zzz
 *.route53docs.com
arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-98a5-443d-a734-800430c80zzz
 nerdzizm.net
arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-2be6-4376-8fa7-bad559525zzz

arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-e7ca-44c5-803e-24d9f2f36zzz

arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-1241-4b71-80b1-090305a62zzz

arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-8709-4568-8c64-f94617c99zzz

arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-a8fa-4a61-98cf-e08ccc0eezzz

arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-fa47-40fe-a714-2d277d3eezzz
 *.route53docs.com
```

- Pour API plus de détails, consultez la section ListCertificatesRéférence des AWS Tools for PowerShell applets de commande.

### `New-ACMCertificate`

L'exemple de code suivant montre comment utiliser`New-ACMCertificate`.

Outils pour PowerShell

Exemple 1 : Crée un nouveau certificat. Le service renvoie ARN le nouveau certificat.

```
New-ACMCertificate -DomainName "www.example.com"
```

Sortie :

```
arn:aws:acm:us-east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
```

Exemple 2 : Crée un nouveau certificat. Le service renvoie ARN le nouveau certificat.

```
New-ACMCertificate -DomainName "www.example.com" -SubjectAlternativeName
 "example.com","www.example.net"
```

Sortie :

```
arn:aws:acm:us-east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
```

- Pour API plus de détails, consultez la section RequestCertificateRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-ACMCertificate`

L'exemple de code suivant montre comment utiliser`Remove-ACMCertificate`.

Outils pour PowerShell

Exemple 1 : Supprime le certificat identifié par la clé privée fournie ARN et la clé privée associée. L'applet de commande demandera une confirmation avant de continuer ; ajoutez le commutateur -Force pour supprimer la confirmation.

```
Remove-ACMCertificate -CertificateArn "arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
```

- Pour API plus de détails, consultez la section DeleteCertificateRéférence des AWS Tools for PowerShell applets de commande.

### `Send-ACMValidationEmail`

L'exemple de code suivant montre comment utiliser`Send-ACMValidationEmail`.

Outils pour PowerShell

Exemple 1 : demande que l'e-mail de validation de la propriété du domaine pour « www.exemple.com » soit envoyé. Si le $ de votre shell ConfirmPreference est défini sur « Moyen » ou inférieur, l'applet de commande vous demandera une confirmation avant de continuer. Ajoutez le commutateur -Force pour supprimer les demandes de confirmation.

```
$params = @{
    CertificateArn="arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
    Domain="www.example.com"
    ValidationDomain="example.com"
}
Send-ACMValidationEmail @params
```

- Pour API plus de détails, consultez la section ResendValidationEmailRéférence des AWS Tools for PowerShell applets de commande.

# Exemples d'Application Auto Scaling utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide de l' AWS Tools for PowerShell application Application Auto Scaling.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `Add-AASScalableTarget`

L'exemple de code suivant montre comment utiliser`Add-AASScalableTarget`.

Outils pour PowerShell

Exemple 1 : cette applet de commande enregistre ou met à jour une cible évolutive. Une cible évolutive est une ressource qu'Application Auto Scaling peut étendre et intégrer.

```
Add-AASScalableTarget -ServiceNamespace AppStream -ResourceId fleet/MyFleet -
ScalableDimension appstream:fleet:DesiredCapacity -MinCapacity 2 -MaxCapacity 10
```

- Pour API plus de détails, consultez la section RegisterScalableTargetRéférence des AWS Tools for PowerShell applets de commande.

### `Get-AASScalableTarget`

L'exemple de code suivant montre comment utiliser`Get-AASScalableTarget`.

Outils pour PowerShell

Exemple 1 : Cet exemple fournira des informations sur les cibles évolutives de l'application Autoscaling dans l'espace de noms spécifié.

```
Get-AASScalableTarget -ServiceNamespace "AppStream"
```

Sortie :

```
CreationTime      : 11/7/2019 2:30:03 AM
MaxCapacity       : 5
MinCapacity       : 1
ResourceId        : fleet/Test
RoleARN           : arn:aws:iam::012345678912:role/aws-
service-role/appstream.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_AppStreamFleet
ScalableDimension : appstream:fleet:DesiredCapacity
ServiceNamespace  : appstream
SuspendedState    : Amazon.ApplicationAutoScaling.Model.SuspendedState
```

- Pour API plus de détails, consultez la section DescribeScalableTargetsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-AASScalingActivity`

L'exemple de code suivant montre comment utiliser`Get-AASScalingActivity`.

Outils pour PowerShell

Exemple 1 : fournit des informations descriptives sur les activités de dimensionnement menées dans l'espace de noms spécifié au cours des six semaines précédentes.

```
Get-AASScalingActivity -ServiceNamespace AppStream
```

Sortie :

```
ActivityId        : 2827409f-b639-4cdb-a957-8055d5d07434
```

```
Cause             : monitor alarm Appstream2-MyFleet-default-scale-in-Alarm in state
 ALARM triggered policy default-scale-in
Description       : Setting desired capacity to 2.
Details           :
EndTime           : 12/14/2019 11:32:49 AM
ResourceId        : fleet/MyFleet
ScalableDimension : appstream:fleet:DesiredCapacity
ServiceNamespace  : appstream
StartTime         : 12/14/2019 11:32:14 AM
StatusCode        : Successful
StatusMessage     : Successfully set desired capacity to 2. Change successfully
 fulfilled by appstream.
```

- Pour API plus de détails, consultez la section DescribeScalingActivitiesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-AASScalingPolicy`

L'exemple de code suivant montre comment utiliser`Get-AASScalingPolicy`.

Outils pour PowerShell

Exemple 1 : Cette applet de commande décrit les politiques de dimensionnement d'Application Auto Scaling pour l'espace de noms de service spécifié.

```
Get-AASScalingPolicy -ServiceNamespace AppStream
```

Sortie :

```
Alarms                                  : {Appstream2-LabFleet-default-scale-out-
Alarm}
CreationTime                            : 9/3/2019 2:48:15 AM
PolicyARN                               : arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:5659b069-b5cd-4af1-9f7f-3e956d36233e:resource/
appstream/fleet/LabFleet:
                                            policyName/default-scale-out
PolicyName                              : default-scale-out
PolicyType                              : StepScaling
ResourceId                              : fleet/LabFleet
ScalableDimension                       : appstream:fleet:DesiredCapacity
ServiceNamespace                        : appstream
```

```
StepScalingPolicyConfiguration          :
 Amazon.ApplicationAutoScaling.Model.StepScalingPolicyConfiguration
TargetTrackingScalingPolicyConfiguration :

Alarms                                  : {Appstream2-LabFleet-default-scale-in-
Alarm}
CreationTime                            : 9/3/2019 2:48:15 AM
PolicyARN                               : arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:5659b069-b5cd-4af1-9f7f-3e956d36233e:resource/
appstream/fleet/LabFleet:
                                            policyName/default-scale-in
PolicyName                              : default-scale-in
PolicyType                              : StepScaling
ResourceId                              : fleet/LabFleet
ScalableDimension                       : appstream:fleet:DesiredCapacity
ServiceNamespace                        : appstream
StepScalingPolicyConfiguration          :
 Amazon.ApplicationAutoScaling.Model.StepScalingPolicyConfiguration
TargetTrackingScalingPolicyConfiguration :
```

- Pour API plus de détails, consultez la section DescribeScalingPoliciesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-AASScheduledAction`

L'exemple de code suivant montre comment utiliser`Get-AASScheduledAction`.

Outils pour PowerShell

Exemple 1 : Cette applet de commande répertorie les actions planifiées pour votre groupe Auto Scaling qui n'ont pas été exécutées ou qui n'ont pas atteint leur heure de fin.

```
Get-AASScheduledAction -ServiceNamespace AppStream
```

Sortie :

```
CreationTime         : 12/22/2019 9:25:52 AM
EndTime              : 1/1/0001 12:00:00 AM
ResourceId           : fleet/MyFleet
ScalableDimension    : appstream:fleet:DesiredCapacity
ScalableTargetAction : Amazon.ApplicationAutoScaling.Model.ScalableTargetAction
```

```
Schedule             : cron(0 0 8 ? * MON-FRI *)
ScheduledActionARN   : arn:aws:autoscaling:us-
west-2:012345678912:scheduledAction:4897ca24-3caa-4bf1-8484-851a089b243c:resource/
appstream/fleet/MyFleet:scheduledActionName
                       /WeekDaysFleetScaling
ScheduledActionName  : WeekDaysFleetScaling
ServiceNamespace     : appstream
StartTime            : 1/1/0001 12:00:00 AM
```

- Pour API plus de détails, consultez la section DescribeScheduledActionsRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-AASScalableTarget`

L'exemple de code suivant montre comment utiliser`Remove-AASScalableTarget`.

Outils pour PowerShell

Exemple 1 : cette applet de commande annule l'enregistrement d'une cible évolutive Application Auto Scaling. Le désenregistrement d'une cible évolutive supprime les politiques de dimensionnement qui lui sont associées.

```
Remove-AASScalableTarget -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity -ServiceNamespace AppStream
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-AASScalableTarget (DeregisterScalableTarget)" on
 target "fleet/MyFleet".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Pour API plus de détails, consultez la section DeregisterScalableTargetRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-AASScalingPolicy`

L'exemple de code suivant montre comment utiliser`Remove-AASScalingPolicy`.

Outils pour PowerShell

Exemple 1 : Cette applet de commande supprime la politique de dimensionnement spécifiée pour une cible évolutive Application Auto Scaling.

```
Remove-AASScalingPolicy -ServiceNamespace AppStream -PolicyName "default-scale-out"
 -ResourceId fleet/Test -ScalableDimension appstream:fleet:DesiredCapacity
```

- Pour API plus de détails, consultez la section DeleteScalingPolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-AASScheduledAction`

L'exemple de code suivant montre comment utiliser`Remove-AASScheduledAction`.

Outils pour PowerShell

Exemple 1 : Cette applet de commande supprime l'action planifiée spécifiée pour une cible évolutive Application Auto Scaling.

```
Remove-AASScheduledAction -ServiceNamespace AppStream -ScheduledActionName
 WeekDaysFleetScaling -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-AASScheduledAction (DeleteScheduledAction)" on
 target "WeekDaysFleetScaling".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Pour API plus de détails, consultez la section DeleteScheduledActionRéférence des AWS Tools for PowerShell applets de commande.

### `Set-AASScalingPolicy`

L'exemple de code suivant montre comment utiliser`Set-AASScalingPolicy`.

Outils pour PowerShell

Exemple 1 : Cette applet de commande crée ou met à jour une politique pour une cible évolutive Application Auto Scaling. Chaque cible évolutive est identifiée par un espace de noms de service, un ID de ressource et une dimension évolutive.

```
Set-AASScalingPolicy -ServiceNamespace AppStream -PolicyName ASFleetScaleInPolicy
 -PolicyType StepScaling  -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity -StepScalingPolicyConfiguration_AdjustmentType
 ChangeInCapacity -StepScalingPolicyConfiguration_Cooldown 360
 -StepScalingPolicyConfiguration_MetricAggregationType Average -
StepScalingPolicyConfiguration_StepAdjustments @{ScalingAdjustment = -1;
 MetricIntervalUpperBound = 0}
```

Sortie :

```
Alarms    PolicyARN
------    ---------
{}        arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:4897ca24-3caa-4bf1-8484-851a089b243c:resource/
appstream/fleet/MyFleet:policyName/ASFleetScaleInPolicy
```

- Pour API plus de détails, consultez la section PutScalingPolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Set-AASScheduledAction`

L'exemple de code suivant montre comment utiliser`Set-AASScheduledAction`.

Outils pour PowerShell

Exemple 1 : Cette applet de commande crée ou met à jour une action planifiée pour une cible évolutive Application Auto Scaling. Chaque cible évolutive est identifiée par un espace de noms de service, un ID de ressource et une dimension évolutive.

```
Set-AASScheduledAction -ServiceNamespace AppStream -ResourceId fleet/
MyFleet -Schedule "cron(0 0 8 ? * MON-FRI *)" -ScalableDimension
 appstream:fleet:DesiredCapacity -ScheduledActionName WeekDaysFleetScaling -
ScalableTargetAction_MinCapacity 5 -ScalableTargetAction_MaxCapacity 10
```

- Pour API plus de détails, consultez la section PutScheduledActionRéférence des AWS Tools for PowerShell applets de commande.

# AppStream Exemples 2.0 utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide de AWS Tools for PowerShell with AppStream 2.0.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `Add-APSResourceTag`

L'exemple de code suivant montre comment utiliser`Add-APSResourceTag`.

Outils pour PowerShell

Exemple 1 : cet exemple ajoute une étiquette de ressource à la AppStream ressource

```
Add-APSResourceTag -ResourceArn arn:aws:appstream:us-east-1:123456789012:stack/
SessionScriptTest -Tag @{StackState='Test'} -Select ^Tag
```

Sortie :

```
Name                           Value
----                           -----
StackState                     Test
```

- Pour API plus de détails, consultez la section TagResourceRéférence des AWS Tools for PowerShell applets de commande.

### `Copy-APSImage`

L'exemple de code suivant montre comment utiliser`Copy-APSImage`.

Outils pour PowerShell

Exemple 1 : Cet exemple copie une image dans une autre région

```
Copy-APSImage -DestinationImageName TestImageCopy -DestinationRegion us-west-2 -
SourceImageName Powershell
```

Sortie :

```
TestImageCopy
```

- Pour API plus de détails, consultez la section CopyImageRéférence des AWS Tools for PowerShell applets de commande.

### `Disable-APSUSer`

L'exemple de code suivant montre comment utiliser`Disable-APSUSer`.

Outils pour PowerShell

Exemple 1 : Cet exemple désactive un utilisateur dans USERPOOL

```
Disable-APSUser -AuthenticationType USERPOOL -UserName TestUser@lab.com
```

- Pour API plus de détails, consultez la section DisableUserRéférence des AWS Tools for PowerShell applets de commande.

### `Enable-APSUser`

L'exemple de code suivant montre comment utiliser`Enable-APSUser`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet à un utilisateur désactivé de USERPOOL

```
Enable-APSUser -AuthenticationType USERPOOL -UserName TestUser@lab.com
```

- Pour API plus de détails, consultez la section EnableUserRéférence des AWS Tools for PowerShell applets de commande.

### `Get-APSAssociatedFleetList`

L'exemple de code suivant montre comment utiliser`Get-APSAssociatedFleetList`.

Outils pour PowerShell

Exemple 1 : Cet exemple affiche le parc associé à une pile

```
Get-APSAssociatedFleetList -StackName PowershellStack
```

Sortie :

```
PowershellFleet
```

- Pour API plus de détails, consultez la section ListAssociatedFleetsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-APSAssociatedStackList`

L'exemple de code suivant montre comment utiliser`Get-APSAssociatedStackList`.

Outils pour PowerShell

Exemple 1 : Cet exemple affiche la pile associée à une flotte

```
Get-APSAssociatedStackList -FleetName PowershellFleet
```

Sortie :

```
PowershellStack
```

- Pour API plus de détails, consultez la section ListAssociatedStacksRéférence des AWS Tools for PowerShell applets de commande.

### `Get-APSDirectoryConfigList`

L'exemple de code suivant montre comment utiliser`Get-APSDirectoryConfigList`.

Outils pour PowerShell

Exemple 1 : Cet exemple affiche les configurations de répertoire créées dans AppStream

```
Get-APSDirectoryConfigList | Select DirectoryName,
 OrganizationalUnitDistinguishedNames, CreatedTime
```

Sortie :

```
DirectoryName OrganizationalUnitDistinguishedNames CreatedTime
------------- ------------------------------------ -----------
Test.com      {OU=AppStream,DC=Test,DC=com}    9/6/2019 10:56:40 AM
contoso.com   {OU=AppStream,OU=contoso,DC=contoso,DC=com}  8/9/2019 9:08:50 AM
```

- Pour API plus de détails, consultez la section DescribeDirectoryConfigsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-APSFleetList`

L'exemple de code suivant montre comment utiliser`Get-APSFleetList`.

Outils pour PowerShell

Exemple 1 : Cet exemple affiche les détails d'une flotte

```
Get-APSFleetList -Name Test
```

Sortie :

```
Arn                           : arn:aws:appstream:us-east-1:1234567890:fleet/Test
ComputeCapacityStatus         : Amazon.AppStream.Model.ComputeCapacityStatus
CreatedTime                   : 9/12/2019 5:00:45 PM
Description                   : Test
DisconnectTimeoutInSeconds    : 900
DisplayName                   : Test
DomainJoinInfo                :
EnableDefaultInternetAccess   : False
FleetErrors                   : {}
FleetType                     : ON_DEMAND
IamRoleArn                    :
IdleDisconnectTimeoutInSeconds : 900
ImageArn                      : arn:aws:appstream:us-east-1:1234567890:image/Test
```

```
ImageName                    : Test
InstanceType                 : stream.standard.medium
MaxUserDurationInSeconds     : 57600
Name                         : Test
State                        : STOPPED
VpcConfig                    : Amazon.AppStream.Model.VpcConfig
```

- Pour API plus de détails, consultez la section DescribeFleetsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-APSImageBuilderList`

L'exemple de code suivant montre comment utiliser`Get-APSImageBuilderList`.

Outils pour PowerShell

Exemple 1 : Cet exemple affiche les détails d'un ImageBuilder

```
Get-APSImageBuilderList -Name TestImage
```

Sortie :

```
AccessEndpoints             : {}
AppstreamAgentVersion       : 06-19-2019
Arn                         : arn:aws:appstream:us-east-1:1234567890:image-builder/
TestImage
CreatedTime                 : 1/14/2019 4:33:05 AM
Description                 :
DisplayName                 : TestImage
DomainJoinInfo              :
EnableDefaultInternetAccess : False
IamRoleArn                  :
ImageArn                    : arn:aws:appstream:us-east-1::image/Base-Image-
Builder-05-02-2018
ImageBuilderErrors          : {}
InstanceType                : stream.standard.large
Name                        : TestImage
NetworkAccessConfiguration  : Amazon.AppStream.Model.NetworkAccessConfiguration
Platform                    : WINDOWS
State                       : STOPPED
StateChangeReason           :
VpcConfig                   : Amazon.AppStream.Model.VpcConfig
```

- Pour API plus de détails, consultez la section DescribeImageBuildersRéférence des AWS Tools for PowerShell applets de commande.

### `Get-APSImageList`

L'exemple de code suivant montre comment utiliser`Get-APSImageList`.

Outils pour PowerShell

Exemple 1 : Cet exemple affiche des AppStream images privées

```
Get-APSImageList -Type PRIVATE | select DisplayName, ImageBuilderName, Visibility,
 arn
```

Sortie :

```
DisplayName          ImageBuilderName      Visibility Arn
-----------          ----------------      ---------- ---
OfficeApps           OfficeApps            PRIVATE    arn:aws:appstream:us-
east-1:123456789012:image/OfficeApps
SessionScriptV2      SessionScriptTest     PRIVATE    arn:aws:appstream:us-
east-1:123456789012:image/SessionScriptV2
```

- Pour API plus de détails, consultez la section DescribeImagesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-APSImagePermission`

L'exemple de code suivant montre comment utiliser`Get-APSImagePermission`.

Outils pour PowerShell

Exemple 1 : Cet exemple affiche les autorisations relatives aux images sur une AppStream image partagée

```
Get-APSImagePermission -Name Powershell  | select SharedAccountId,
 @{n="AllowFleet";e={$_.ImagePermissions.AllowFleet}},
 @{n="AllowImageBuilder";e={$_.ImagePermissions.AllowImageBuilder}}
```

Sortie :

```
SharedAccountId AllowFleet AllowImageBuilder
--------------- ---------- -----------------
123456789012          True              True
```

- Pour API plus de détails, consultez la section DescribeImagePermissionsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-APSSessionList`

L'exemple de code suivant montre comment utiliser`Get-APSSessionList`.

Outils pour PowerShell

Exemple 1 : Cet exemple affiche la liste des sessions d'une flotte

```
Get-APSSessionList -FleetName PowershellFleet -StackName PowershellStack
```

Sortie :

```
AuthenticationType         : API
ConnectionState            : CONNECTED
FleetName                  : PowershellFleet
Id                         : d8987c70-4394-4324-a396-2d485c26f2a2
MaxExpirationTime          : 12/27/2019 4:54:07 AM
NetworkAccessConfiguration : Amazon.AppStream.Model.NetworkAccessConfiguration
StackName                  : PowershellStack
StartTime                  : 12/26/2019 12:54:12 PM
State                      : ACTIVE
UserId                     : Test
```

- Pour API plus de détails, consultez la section DescribeSessionsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-APSStackList`

L'exemple de code suivant montre comment utiliser`Get-APSStackList`.

Outils pour PowerShell

Exemple 1 : Cet exemple affiche la liste des AppStream Stack

```
Get-APSStackList | Select DisplayName, Arn, CreatedTime
```

Sortie :

```
DisplayName                  Arn
                       CreatedTime
-----------                  ---
                       -----------
PowershellStack              arn:aws:appstream:us-east-1:123456789012:stack/
PowershellStack              4/24/2019 8:49:29 AM
SessionScriptTest            arn:aws:appstream:us-east-1:123456789012:stack/
SessionScriptTest            9/12/2019 3:23:12 PM
```

- Pour API plus de détails, consultez la section DescribeStacksRéférence des AWS Tools for PowerShell applets de commande.

### `Get-APSTagsForResourceList`

L'exemple de code suivant montre comment utiliser`Get-APSTagsForResourceList`.

Outils pour PowerShell

Exemple 1 : cet exemple affiche les balises d'une AppStream ressource

```
Get-APSTagsForResourceList -ResourceArn arn:aws:appstream:us-
east-1:123456789012:stack/SessionScriptTest
```

Sortie :

```
Key        Value
---        -----
StackState Test
```

- Pour API plus de détails, consultez la section ListTagsForResourceRéférence des AWS Tools for PowerShell applets de commande.

### `Get-APSUsageReportSubscription`

L'exemple de code suivant montre comment utiliser`Get-APSUsageReportSubscription`.

Outils pour PowerShell

Exemple 1 : cet exemple affiche les détails AppStreamUsageReport de configuration

```
Get-APSUsageReportSubscription
```

Sortie :

```
LastGeneratedReportDate S3BucketName                                    Schedule
 SubscriptionErrors
----------------------- ------------                                    --------
 ------------------
1/1/0001 12:00:00 AM    appstream-logs-us-east-1-123456789012-sik1hnxe DAILY    {}
```

- Pour API plus de détails, consultez la section DescribeUsageReportSubscriptionsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-APSUser`

L'exemple de code suivant montre comment utiliser`Get-APSUser`.

Outils pour PowerShell

Exemple 1 : Cet exemple affiche la liste des utilisateurs dont le statut est activé

```
Get-APSUser -AuthenticationType USERPOOL | Select-Object UserName,
 AuthenticationType, Enabled
```

Sortie :

```
UserName                  AuthenticationType Enabled
--------                  ------------------ -------
foo1@contoso.com USERPOOL              True
foo2@contoso.com        USERPOOL              True
foo3@contoso.com    USERPOOL             True
foo4@contoso.com   USERPOOL             True
foo5@contoso.com         USERPOOL              True
```

- Pour API plus de détails, consultez la section DescribeUsersRéférence des AWS Tools for PowerShell applets de commande.

### `Get-APSUserStackAssociation`

L'exemple de code suivant montre comment utiliser`Get-APSUserStackAssociation`.

Outils pour PowerShell

Exemple 1 : Cet exemple affiche la liste des utilisateurs assignés à une pile

```
Get-APSUserStackAssociation -StackName PowershellStack
```

Sortie :

```
AuthenticationType SendEmailNotification StackName       UserName
------------------ --------------------- ---------       --------
USERPOOL           False                 PowershellStack TestUser1@lab.com
USERPOOL           False                 PowershellStack TestUser2@lab.com
```

- Pour API plus de détails, consultez la section DescribeUserStackAssociationsRéférence des AWS Tools for PowerShell applets de commande.

### `New-APSDirectoryConfig`

L'exemple de code suivant montre comment utiliser`New-APSDirectoryConfig`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une configuration de répertoire dans AppStream

```
New-APSDirectoryConfig -ServiceAccountCredentials_AccountName contoso\ServiceAccount
 -ServiceAccountCredentials_AccountPassword MyPass -DirectoryName contoso.com -
OrganizationalUnitDistinguishedName "OU=AppStream,OU=Contoso,DC=Contoso,DC=com"
```

Sortie :

```
CreatedTime            DirectoryName OrganizationalUnitDistinguishedNames
 ServiceAccountCredentials
-----------            ------------- ------------------------------------
 -------------------------
12/27/2019 11:00:30 AM contoso.com   {OU=AppStream,OU=Contoso,DC=Contoso,DC=com}
 Amazon.AppStream.Model.ServiceAccountCredentials
```

- Pour API plus de détails, consultez la section CreateDirectoryConfigRéférence des AWS Tools for PowerShell applets de commande.

### `New-APSFleet`

L'exemple de code suivant montre comment utiliser`New-APSFleet`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une nouvelle AppStream flotte

```
New-APSFleet -ComputeCapacity_DesiredInstance 1 -InstanceType stream.standard.medium
 -Name TestFleet -DisplayName TestFleet -FleetType ON_DEMAND -
EnableDefaultInternetAccess $True -VpcConfig_SubnetIds "subnet-123ce32","subnet-
a1234cfd" -VpcConfig_SecurityGroupIds sg-4d012a34 -ImageName SessionScriptTest -
Region us-west-2
```

Sortie :

```
Arn                           : arn:aws:appstream:us-west-2:123456789012:fleet/
TestFleet
ComputeCapacityStatus         : Amazon.AppStream.Model.ComputeCapacityStatus
CreatedTime                   : 12/27/2019 11:24:42 AM
Description                   :
DisconnectTimeoutInSeconds    : 900
DisplayName                   : TestFleet
DomainJoinInfo                :
EnableDefaultInternetAccess   : True
FleetErrors                   : {}
FleetType                     : ON_DEMAND
IamRoleArn                    :
IdleDisconnectTimeoutInSeconds : 0
ImageArn                      : arn:aws:appstream:us-west-2:123456789012:image/
SessionScriptTest
ImageName                     : SessionScriptTest
InstanceType                  : stream.standard.medium
MaxUserDurationInSeconds      : 57600
Name                          : TestFleet
State                         : STOPPED
VpcConfig                     : Amazon.AppStream.Model.VpcConfig
```

- Pour API plus de détails, consultez la section CreateFleetRéférence des AWS Tools for PowerShell applets de commande.

### `New-APSImageBuilder`

L'exemple de code suivant montre comment utiliser`New-APSImageBuilder`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un Image Builder dans AppStream

```
New-APSImageBuilder -InstanceType stream.standard.medium -Name TestIB -DisplayName
 TestIB -ImageName AppStream-WinServer2012R2-12-12-2019 -EnableDefaultInternetAccess
 $True -VpcConfig_SubnetId subnet-a1234cfd -VpcConfig_SecurityGroupIds sg-2d012a34 -
Region us-west-2
```

Sortie :

```
AccessEndpoints             : {}
AppstreamAgentVersion       : 12-16-2019
Arn                         : arn:aws:appstream:us-west-2:123456789012:image-
builder/TestIB
CreatedTime                 : 12/27/2019 11:39:24 AM
Description                 :
DisplayName                 : TestIB
DomainJoinInfo              :
EnableDefaultInternetAccess : True
IamRoleArn                  :
ImageArn                    : arn:aws:appstream:us-west-2::image/AppStream-
WinServer2012R2-12-12-2019
ImageBuilderErrors          : {}
InstanceType                : stream.standard.medium
Name                        : TestIB
NetworkAccessConfiguration  :
Platform                    : WINDOWS
State                       : PENDING
StateChangeReason           :
VpcConfig                   : Amazon.AppStream.Model.VpcConfig
```

- Pour API plus de détails, consultez la section CreateImageBuilderRéférence des AWS Tools for PowerShell applets de commande.

### `New-APSImageBuilderStreamingURL`

L'exemple de code suivant montre comment utiliser`New-APSImageBuilderStreamingURL`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un ImageBuilder streaming URL d'une validité de 2 heures

```
New-APSImageBuilderStreamingURL -Name TestIB -Validity 7200 -Region us-west-2
```

Sortie :

```
Expires               StreamingURL
-------               ------------
12/27/2019 1:49:13 PM https://appstream2.us-west-2.aws.amazon.com/authenticate?
parameters=eyJ0eXBlIjoiQURNSU4iLCJleHBpcmVzIjoiMTU3NzQ1NDU1MyIsImF3c0FjY291bnRJZCI6IjM5MzQwM
```

- Pour API plus de détails, consultez la section CreateImageBuilderStreamingURLRéférence des AWS Tools for PowerShell applets de commande.

### `New-APSStack`

L'exemple de code suivant montre comment utiliser`New-APSStack`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une nouvelle AppStream pile

```
New-APSStack -Name TestStack -DisplayName TestStack -ApplicationSettings_Enabled
 $True -ApplicationSettings_SettingsGroup TestStack -Region us-west-2
```

Sortie :

```
AccessEndpoints     : {}
ApplicationSettings : Amazon.AppStream.Model.ApplicationSettingsResponse
Arn                 : arn:aws:appstream:us-west-2:123456789012:stack/TestStack
CreatedTime         : 12/27/2019 12:34:19 PM
Description         :
DisplayName         : TestStack
EmbedHostDomains    : {}
FeedbackURL         :
Name                : TestStack
```

```
RedirectURL         :
StackErrors         : {}
StorageConnectors   : {}
UserSettings        : {Amazon.AppStream.Model.UserSetting,
 Amazon.AppStream.Model.UserSetting, Amazon.AppStream.Model.UserSetting,
 Amazon.AppStream.Model.UserSetting}
```

- Pour API plus de détails, consultez la section CreateStackRéférence des AWS Tools for PowerShell applets de commande.

### `New-APSStreamingURL`

L'exemple de code suivant montre comment utiliser`New-APSStreamingURL`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un streaming URL de Stack

```
New-APSStreamingURL -StackName SessionScriptTest -FleetName SessionScriptNew -UserId
 TestUser
```

Sortie :

```
Expires                 StreamingURL
-------                 ------------
12/27/2019 12:43:37 PM https://appstream2.us-east-1.aws.amazon.com/authenticate?
parameters=eyJ0eXBlIjoiRU5EX1VTRVIiLCJleHBpcmVzIjoiMTU3NzQ1MDYxNyIsImF3c0FjY291bnRJZCI6IjM5M
```

- Pour API plus de détails, consultez la section CreateStreamingURLRéférence des AWS Tools for PowerShell applets de commande.

### `New-APSUsageReportSubscription`

L'exemple de code suivant montre comment utiliser`New-APSUsageReportSubscription`.

Outils pour PowerShell

Exemple 1 : Cet exemple active les rapports AppStream d'utilisation

```
New-APSUsageReportSubscription
```

Sortie :

```
S3BucketName                                    Schedule
------------                                    --------
appstream-logs-us-east-1-123456789012-sik2hnxe DAILY
```

- Pour API plus de détails, consultez la section CreateUsageReportSubscriptionRéférence des AWS Tools for PowerShell applets de commande.

### `New-APSUser`

L'exemple de code suivant montre comment utiliser`New-APSUser`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un utilisateur dans USERPOOL

```
New-APSUser -UserName Test@lab.com -AuthenticationType USERPOOL -FirstName 'kt' -
LastName 'aws' -Select ^UserName
```

Sortie :

```
Test@lab.com
```

- Pour API plus de détails, consultez la section CreateUserRéférence des AWS Tools for PowerShell applets de commande.

### `Register-APSFleet`

L'exemple de code suivant montre comment utiliser`Register-APSFleet`.

Outils pour PowerShell

Exemple 1 : Cet exemple enregistre une flotte avec une pile

```
Register-APSFleet -StackName TestStack -FleetName TestFleet -Region us-west-2
```

- Pour API plus de détails, consultez la section AssociateFleetRéférence des AWS Tools for PowerShell applets de commande.

### `Register-APSUserStackBatch`

L'exemple de code suivant montre comment utiliser`Register-APSUserStackBatch`.

Outils pour PowerShell

Exemple 1 : Cet exemple attribue une pile à un utilisateur dans USERPOOL

```
Register-APSUserStackBatch -UserStackAssociation
 @{AuthenticationType="USERPOOL";SendEmailNotification=
$False;StackName="PowershellStack";UserName="TestUser1@lab.com"}
```

- Pour API plus de détails, consultez la section BatchAssociateUserStackRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-APSDirectoryConfig`

L'exemple de code suivant montre comment utiliser`Remove-APSDirectoryConfig`.

Outils pour PowerShell

Exemple 1 : cet exemple supprime la configuration du AppStream répertoire

```
Remove-APSDirectoryConfig -DirectoryName contoso.com
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSDirectoryConfig (DeleteDirectoryConfig)" on
 target "contoso.com".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): A
```

- Pour API plus de détails, consultez la section DeleteDirectoryConfigRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-APSFleet`

L'exemple de code suivant montre comment utiliser`Remove-APSFleet`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime et supprime une flotte AppStream

```
Remove-APSFleet -Name TestFleet -Region us-west-2
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSFleet (DeleteFleet)" on target "TestFleet".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): A
```

- Pour API plus de détails, consultez la section DeleteFleetRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-APSImage`

L'exemple de code suivant montre comment utiliser`Remove-APSImage`.

Outils pour PowerShell

Exemple 1 : cet exemple supprime une image

```
Remove-APSImage -Name TestImage -Region us-west-2
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSImage (DeleteImage)" on target "TestImage".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): A

Applications               : {}
AppstreamAgentVersion      : LATEST
Arn                        : arn:aws:appstream:us-west-2:123456789012:image/
TestImage
BaseImageArn               :
CreatedTime                : 12/27/2019 1:34:10 PM
```

```
Description                 :
DisplayName                 : TestImage
ImageBuilderName            :
ImageBuilderSupported       : True
ImagePermissions            :
Name                        : TestImage
Platform                    : WINDOWS
PublicBaseImageReleasedDate : 6/12/2018 12:00:00 AM
State                       : AVAILABLE
StateChangeReason           :
Visibility                  : PRIVATE
```

- Pour API plus de détails, consultez la section DeleteImageRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-APSImageBuilder`

L'exemple de code suivant montre comment utiliser`Remove-APSImageBuilder`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime un ImageBuilder

```
Remove-APSImageBuilder -Name TestIB -Region us-west-2
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSImageBuilder (DeleteImageBuilder)" on target
 "TestIB".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): A

AccessEndpoints             : {}
AppstreamAgentVersion       : 12-16-2019
Arn                         : arn:aws:appstream:us-west-2:123456789012:image-
builder/TestIB
CreatedTime                 : 12/27/2019 11:39:24 AM
Description                 :
DisplayName                 : TestIB
DomainJoinInfo              :
```

```
EnableDefaultInternetAccess : True
IamRoleArn                  :
ImageArn                    : arn:aws:appstream:us-west-2::image/AppStream-
WinServer2012R2-12-12-2019
ImageBuilderErrors          : {}
InstanceType                : stream.standard.medium
Name                        : TestIB
NetworkAccessConfiguration  : Amazon.AppStream.Model.NetworkAccessConfiguration
Platform                    : WINDOWS
State                       : DELETING
StateChangeReason           :
VpcConfig                   : Amazon.AppStream.Model.VpcConfig
```

- Pour API plus de détails, consultez la section DeleteImageBuilderRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-APSImagePermission`

L'exemple de code suivant montre comment utiliser`Remove-APSImagePermission`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime les autorisations d'une image

```
Remove-APSImagePermission -Name Powershell -SharedAccountId 123456789012
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSImagePermission (DeleteImagePermissions)" on
 target "Powershell".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): A
```

- Pour API plus de détails, consultez la section DeleteImagePermissionsRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-APSResourceTag`

L'exemple de code suivant montre comment utiliser`Remove-APSResourceTag`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime une balise de ressource de la AppStream ressource

```
Remove-APSResourceTag -ResourceArn arn:aws:appstream:us-east-1:123456789012:stack/
SessionScriptTest -TagKey StackState
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSResourceTag (UntagResource)" on target
 "arn:aws:appstream:us-east-1:123456789012:stack/SessionScriptTest".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): A
```

- Pour API plus de détails, consultez la section UntagResourceRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-APSStack`

L'exemple de code suivant montre comment utiliser`Remove-APSStack`.

Outils pour PowerShell

Exemple 1 : cet exemple supprime une pile

```
Remove-APSStack -Name TestStack -Region us-west-2
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSStack (DeleteStack)" on target "TestStack".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): A
```

- Pour API plus de détails, consultez la section DeleteStackRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-APSUsageReportSubscription`

L'exemple de code suivant montre comment utiliser`Remove-APSUsageReportSubscription`.

Outils pour PowerShell

Exemple 1 : cet exemple désactive l'abonnement au rapport AppStream d'utilisation

```
Remove-APSUsageReportSubscription
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSUsageReportSubscription
 (DeleteUsageReportSubscription)" on target "".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): A
```

- Pour API plus de détails, consultez la section DeleteUsageReportSubscriptionRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-APSUser`

L'exemple de code suivant montre comment utiliser`Remove-APSUser`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime un utilisateur de USERPOOL

```
Remove-APSUser -UserName TestUser@lab.com -AuthenticationType USERPOOL
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSUser (DeleteUser)" on target "TestUser@lab.com".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): A
```

- Pour API plus de détails, consultez la section DeleteUserRéférence des AWS Tools for PowerShell applets de commande.

### `Revoke-APSSession`

L'exemple de code suivant montre comment utiliser`Revoke-APSSession`.

Outils pour PowerShell

Exemple 1 : Cet exemple révoque une session dans le AppStream parc

```
Revoke-APSSession -SessionId 6cd2f9a3-f948-4aa1-8014-8a7dcde14877
```

- Pour API plus de détails, consultez la section ExpireSessionRéférence des AWS Tools for PowerShell applets de commande.

### `Start-APSFleet`

L'exemple de code suivant montre comment utiliser`Start-APSFleet`.

Outils pour PowerShell

Exemple 1 : Cet exemple démarre une flotte

```
Start-APSFleet -Name PowershellFleet
```

- Pour API plus de détails, consultez la section StartFleetRéférence des AWS Tools for PowerShell applets de commande.

### `Start-APSImageBuilder`

L'exemple de code suivant montre comment utiliser`Start-APSImageBuilder`.

Outils pour PowerShell

Exemple 1 : Cet exemple démarre un ImageBuilder

```
Start-APSImageBuilder -Name TestImage
```

Sortie :

```
AccessEndpoints             : {}
AppstreamAgentVersion       : 06-19-2019
Arn                         : arn:aws:appstream:us-east-1:123456789012:image-
builder/TestImage
CreatedTime                 : 1/14/2019 4:33:05 AM
Description                 :
DisplayName                 : TestImage
DomainJoinInfo              :
EnableDefaultInternetAccess : False
IamRoleArn                  :
ImageArn                    : arn:aws:appstream:us-east-1::image/Base-Image-
Builder-05-02-2018
ImageBuilderErrors          : {}
InstanceType                : stream.standard.large
Name                        : TestImage
NetworkAccessConfiguration  : Amazon.AppStream.Model.NetworkAccessConfiguration
Platform                    : WINDOWS
State                       : PENDING
StateChangeReason           :
VpcConfig                   : Amazon.AppStream.Model.VpcConfig
```

- Pour API plus de détails, consultez la section StartImageBuilderRéférence des AWS Tools for PowerShell applets de commande.

### `Stop-APSFleet`

L'exemple de code suivant montre comment utiliser`Stop-APSFleet`.

Outils pour PowerShell

Exemple 1 : Cet exemple arrête une flotte

```
Stop-APSFleet -Name PowershellFleet
```

- Pour API plus de détails, consultez la section StopFleetRéférence des AWS Tools for PowerShell applets de commande.

### `Stop-APSImageBuilder`

L'exemple de code suivant montre comment utiliser`Stop-APSImageBuilder`.

Outils pour PowerShell

Exemple 1 : Cet exemple arrête un ImageBuilder

```
Stop-APSImageBuilder -Name TestImage
```

Sortie :

```
AccessEndpoints             : {}
AppstreamAgentVersion       : 06-19-2019
Arn                         : arn:aws:appstream:us-east-1:123456789012:image-
builder/TestImage
CreatedTime                 : 1/14/2019 4:33:05 AM
Description                 :
DisplayName                 : TestImage
DomainJoinInfo              :
EnableDefaultInternetAccess : False
IamRoleArn                  :
ImageArn                    : arn:aws:appstream:us-east-1::image/Base-Image-
Builder-05-02-2018
ImageBuilderErrors          : {}
InstanceType                : stream.standard.large
Name                        : TestImage
NetworkAccessConfiguration  : Amazon.AppStream.Model.NetworkAccessConfiguration
Platform                    : WINDOWS
State                       : STOPPING
StateChangeReason           :
VpcConfig                   : Amazon.AppStream.Model.VpcConfig
```

- Pour API plus de détails, consultez la section StopImageBuilderRéférence des AWS Tools for PowerShell applets de commande.

### `Unregister-APSFleet`

L'exemple de code suivant montre comment utiliser`Unregister-APSFleet`.

Outils pour PowerShell

Exemple 1 : Cet exemple annule l'enregistrement d'une flotte de Stack

```
Unregister-APSFleet -StackName TestStack -FleetName TestFleet -Region us-west-2
```

- Pour API plus de détails, consultez la section DisassociateFleetRéférence des AWS Tools for PowerShell applets de commande.

### `Unregister-APSUserStackBatch`

L'exemple de code suivant montre comment utiliser`Unregister-APSUserStackBatch`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime un utilisateur d'une pile assignée

```
Unregister-APSUserStackBatch -UserStackAssociation
 @{AuthenticationType="USERPOOL";SendEmailNotification=
$False;StackName="PowershellStack";UserName="TestUser1@lab.com"}
```

- Pour API plus de détails, consultez la section BatchDisassociateUserStackRéférence des AWS Tools for PowerShell applets de commande.

### `Update-APSDirectoryConfig`

L'exemple de code suivant montre comment utiliser`Update-APSDirectoryConfig`.

Outils pour PowerShell

Exemple 1 : Cet exemple met à jour la configuration du répertoire créée dans AppStream

```
Update-APSDirectoryConfig -ServiceAccountCredentials_AccountName contoso
\ServiceAccount -ServiceAccountCredentials_AccountPassword MyPass@1$@#
 -DirectoryName contoso.com -OrganizationalUnitDistinguishedName
 "OU=AppStreamNew,OU=Contoso,DC=Contoso,DC=com"
```

Sortie :

```
CreatedTime           DirectoryName OrganizationalUnitDistinguishedNames
 ServiceAccountCredentials
-----------           ------------- ------------------------------------
 -------------------------
12/27/2019 3:50:02 PM contoso.com   {OU=AppStreamNew,OU=Contoso,DC=Contoso,DC=com}
 Amazon.AppStream.Model.ServiceAccountCredentials
```

- Pour API plus de détails, consultez la section UpdateDirectoryConfigRéférence des AWS Tools for PowerShell applets de commande.

### `Update-APSFleet`

L'exemple de code suivant montre comment utiliser`Update-APSFleet`.

Outils pour PowerShell

Exemple 1 : cet exemple met à jour les propriétés d'une flotte

```
Update-APSFleet -Name PowershellFleet -EnableDefaultInternetAccess $True -
DisconnectTimeoutInSecond 950
```

Sortie :

```
Arn                           : arn:aws:appstream:us-east-1:123456789012:fleet/
PowershellFleet
ComputeCapacityStatus         : Amazon.AppStream.Model.ComputeCapacityStatus
CreatedTime                   : 4/24/2019 8:39:41 AM
Description                   : PowershellFleet
DisconnectTimeoutInSeconds    : 950
DisplayName                   : PowershellFleet
DomainJoinInfo                :
EnableDefaultInternetAccess   : True
FleetErrors                   : {}
FleetType                     : ON_DEMAND
IamRoleArn                    :
IdleDisconnectTimeoutInSeconds : 900
ImageArn                      : arn:aws:appstream:us-east-1:123456789012:image/
Powershell
ImageName                     : Powershell
InstanceType                  : stream.standard.medium
MaxUserDurationInSeconds      : 57600
Name                          : PowershellFleet
State                         : STOPPED
VpcConfig                     : Amazon.AppStream.Model.VpcConfig
```

- Pour API plus de détails, consultez la section UpdateFleetRéférence des AWS Tools for PowerShell applets de commande.

### `Update-APSImagePermission`

L'exemple de code suivant montre comment utiliser`Update-APSImagePermission`.

Outils pour PowerShell

Exemple 1 : Cet exemple partage une AppStream image avec un autre compte

```
Update-APSImagePermission -Name Powershell -SharedAccountId 123456789012 -
ImagePermissions_AllowFleet $True -ImagePermissions_AllowImageBuilder $True
```

- Pour API plus de détails, consultez la section UpdateImagePermissionsRéférence des AWS Tools for PowerShell applets de commande.

### `Update-APSStack`

L'exemple de code suivant montre comment utiliser`Update-APSStack`.

Outils pour PowerShell

Exemple 1 : cet exemple met à jour (active) la persistance des paramètres de l'application et les dossiers de base sur une pile

```
Update-APSStack -Name PowershellStack -ApplicationSettings_Enabled $True
 -ApplicationSettings_SettingsGroup PowershellStack -StorageConnector
 @{ConnectorType="HOMEFOLDERS"}
```

Sortie :

```
AccessEndpoints     : {}
ApplicationSettings : Amazon.AppStream.Model.ApplicationSettingsResponse
Arn                 : arn:aws:appstream:us-east-1:123456789012:stack/PowershellStack
CreatedTime         : 4/24/2019 8:49:29 AM
Description         : PowershellStack
DisplayName         : PowershellStack
EmbedHostDomains    : {}
FeedbackURL         :
Name                : PowershellStack
RedirectURL         :
StackErrors         : {}
```

```
StorageConnectors   : {Amazon.AppStream.Model.StorageConnector,
 Amazon.AppStream.Model.StorageConnector}
UserSettings        : {Amazon.AppStream.Model.UserSetting,
 Amazon.AppStream.Model.UserSetting, Amazon.AppStream.Model.UserSetting,
 Amazon.AppStream.Model.UserSetting}
```

- Pour API plus de détails, consultez la section UpdateStackRéférence des AWS Tools for PowerShell applets de commande.

# Exemples d'Aurora utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS Tools for PowerShell aide d'Aurora.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `Get-RDSOrderableDBInstanceOption`

L'exemple de code suivant montre comment utiliser`Get-RDSOrderableDBInstanceOption`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie les versions du moteur de base de données qui prennent en charge une classe d'instance de base de données spécifique dans un Région AWS.

```
$params = @{
  Engine = 'aurora-postgresql'
  DBInstanceClass = 'db.r5.large'
  Region = 'us-east-1'
}
```

```
Get-RDSOrderableDBInstanceOption @params
```

Exemple 2 : Cet exemple répertorie les classes d'instances de base de données prises en charge pour une version de moteur de base de données spécifique dans un Région AWS.

```
$params = @{
  Engine = 'aurora-postgresql'
  EngineVersion = '13.6'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

- Pour API plus de détails, consultez la section DescribeOrderableDBInstanceOptionsRéférence des AWS Tools for PowerShell applets de commande.

# Exemples d'Auto Scaling utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide de AWS Tools for PowerShell with Auto Scaling.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `Add-ASLoadBalancer`

L'exemple de code suivant montre comment utiliser`Add-ASLoadBalancer`.

Outils pour PowerShell

Exemple 1 : Cet exemple attache l'équilibreur de charge spécifié au groupe Auto Scaling spécifié.

```
Add-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- Pour API plus de détails, consultez la section AttachLoadBalancersRéférence des AWS Tools for PowerShell applets de commande.

### `Complete-ASLifecycleAction`

L'exemple de code suivant montre comment utiliser`Complete-ASLifecycleAction`.

Outils pour PowerShell

Exemple 1 : Cet exemple complète l'action du cycle de vie spécifiée.

```
Complete-ASLifecycleAction -LifecycleHookName myLifecycleHook -
AutoScalingGroupName my-asg -LifecycleActionResult CONTINUE -LifecycleActionToken
 bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- Pour API plus de détails, consultez la section CompleteLifecycleActionRéférence des AWS Tools for PowerShell applets de commande.

### `Disable-ASMetricsCollection`

L'exemple de code suivant montre comment utiliser`Disable-ASMetricsCollection`.

Outils pour PowerShell

Exemple 1 : Cet exemple désactive la surveillance des métriques spécifiées pour le groupe Auto Scaling spécifié.

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg -Metric @("GroupMinSize",
 "GroupMaxSize")
```

Exemple 2 : Cet exemple désactive la surveillance de toutes les métriques pour le groupe Auto Scaling spécifié.

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg
```

- Pour API plus de détails, consultez la section DisableMetricsCollectionRéférence des AWS Tools for PowerShell applets de commande.

### `Dismount-ASInstance`

L'exemple de code suivant montre comment utiliser`Dismount-ASInstance`.

Outils pour PowerShell

Exemple 1 : Cet exemple détache l'instance spécifiée du groupe Auto Scaling spécifié et diminue la capacité souhaitée afin qu'Auto Scaling ne lance pas d'instance de remplacement.

```
Dismount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

Sortie :

```
ActivityId           : 06733445-ce94-4039-be1b-b9f1866e276e
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:34:59Z instance i-93633f9b was detached in
 response to a user request, shrinking
                       the capacity from 2 to 1.
Description          : Detaching EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/20/2015 2:34:59 PM
StatusCode           : InProgress
StatusMessage        :
```

Exemple 2 : Cet exemple détache l'instance spécifiée du groupe Auto Scaling spécifié sans diminuer la capacité souhaitée. Auto Scaling lance une instance de remplacement.

```
Dismount-ASInstance -InstanceId i-7bf746a2 -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

Sortie :

```
ActivityId           : f43a3cd4-d38c-4af7-9fe0-d76ec2307b6d
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:34:59Z instance i-7bf746a2 was detached in
 response to a user request.
```

```
Description          : Detaching EC2 instance: i-7bf746a2
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/20/2015 2:34:59 PM
StatusCode           : InProgress
StatusMessage        :
```

- Pour API plus de détails, consultez la section DetachInstancesRéférence des AWS Tools for PowerShell applets de commande.

### `Dismount-ASLoadBalancer`

L'exemple de code suivant montre comment utiliser`Dismount-ASLoadBalancer`.

Outils pour PowerShell

Exemple 1 : Cet exemple détache l'équilibreur de charge spécifié du groupe Auto Scaling spécifié.

```
Dismount-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- Pour API plus de détails, consultez la section DetachLoadBalancersRéférence des AWS Tools for PowerShell applets de commande.

### `Enable-ASMetricsCollection`

L'exemple de code suivant montre comment utiliser`Enable-ASMetricsCollection`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet de surveiller les métriques spécifiées pour le groupe Auto Scaling spécifié.

```
Enable-ASMetricsCollection  -Metric @("GroupMinSize", "GroupMaxSize") -
AutoScalingGroupName my-asg -Granularity 1Minute
```

Exemple 2 : Cet exemple permet de surveiller toutes les métriques pour le groupe Auto Scaling spécifié.

```
Enable-ASMetricsCollection -AutoScalingGroupName my-asg -Granularity 1Minute
```

- Pour API plus de détails, consultez la section EnableMetricsCollectionRéférence des AWS Tools for PowerShell applets de commande.

### `Enter-ASStandby`

L'exemple de code suivant montre comment utiliser`Enter-ASStandby`.

Outils pour PowerShell

Exemple 1 : Cet exemple met l'instance spécifiée en mode veille et diminue la capacité souhaitée afin qu'Auto Scaling ne lance pas d'instance de remplacement.

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

Sortie :

```
ActivityId           : e36a5a54-ced6-4df8-bd19-708e2a59a649
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to
 standby in response to a user request,
                       shrinking the capacity from 2 to 1.
Description          : Moving EC2 instance to Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/22/2015 7:48:06 AM
StatusCode           : InProgress
StatusMessage        :
```

Exemple 2 : Cet exemple met l'instance spécifiée en mode veille sans diminuer la capacité souhaitée. Auto Scaling lance une instance de remplacement.

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

Sortie :

```
ActivityId           : e36a5a54-ced6-4df8-bd19-708e2a59a649
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to
 standby in response to a user request.
Description          : Moving EC2 instance to Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/22/2015 7:48:06 AM
StatusCode           : InProgress
StatusMessage        :
```

- Pour API plus de détails, consultez la section EnterStandbyRéférence des AWS Tools for PowerShell applets de commande.

### `Exit-ASStandby`

L'exemple de code suivant montre comment utiliser`Exit-ASStandby`.

Outils pour PowerShell

Exemple 1 : Cet exemple fait sortir l'instance spécifiée du mode veille.

```
Exit-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

Sortie :

```
ActivityId           : 1833d3e8-e32f-454e-b731-0670ad4c6934
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:51:21Z instance i-95b8484f was moved out of
 standby in response to a user
                       request, increasing the capacity from 1 to 2.
Description          : Moving EC2 instance out of Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 30
StartTime            : 11/22/2015 7:51:21 AM
StatusCode           : PreInService
StatusMessage        :
```

- Pour API plus de détails, consultez la section ExitStandbyRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ASAccountLimit`

L'exemple de code suivant montre comment utiliser`Get-ASAccountLimit`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit les limites de ressources Auto Scaling pour votre AWS compte.

```
Get-ASAccountLimit
```

Sortie :

```
MaxNumberOfAutoScalingGroups    : 20
MaxNumberOfLaunchConfigurations : 100
```

- Pour API plus de détails, consultez la section DescribeAccountLimitsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ASAdjustmentType`

L'exemple de code suivant montre comment utiliser`Get-ASAdjustmentType`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit les types d'ajustement pris en charge par Auto Scaling.

```
Get-ASAdjustmentType
```

Sortie :

```
Type
----
ChangeInCapacity
ExactCapacity
PercentChangeInCapacity
```

- Pour API plus de détails, consultez la section DescribeAdjustmentTypesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ASAutoScalingGroup`

L'exemple de code suivant montre comment utiliser`Get-ASAutoScalingGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie les noms de vos groupes Auto Scaling.

```
Get-ASAutoScalingGroup | format-table -property AutoScalingGroupName
```

Sortie :

```
AutoScalingGroupName
--------------------
my-asg-1
my-asg-2
my-asg-3
my-asg-4
my-asg-5
my-asg-6
```

Exemple 2 : Cet exemple décrit le groupe Auto Scaling spécifié.

```
Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1
```

Sortie :

```
AutoScalingGroupARN     : arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-a11a-7b0acd480
                          f03:autoScalingGroupName/my-asg-1
AutoScalingGroupName    : my-asg-1
AvailabilityZones       : {us-west-2b, us-west-2a}
CreatedTime             : 3/1/2015 9:05:31 AM
DefaultCooldown         : 300
DesiredCapacity         : 2
EnabledMetrics          : {}
HealthCheckGracePeriod  : 300
HealthCheckType         : EC2
```

```
Instances               : {my-lc}
LaunchConfigurationName : my-lc
LoadBalancerNames       : {}
MaxSize                 : 0
MinSize                 : 0
PlacementGroup          :
Status                  :
SuspendedProcesses      : {}
Tags                    : {}
TerminationPolicies     : {Default}
VPCZoneIdentifier       : subnet-e4f33493,subnet-5264e837
```

Exemple 3 : Cet exemple décrit les deux groupes Auto Scaling spécifiés.

```
Get-ASAutoScalingGroup -AutoScalingGroupName @("my-asg-1", "my-asg-2")
```

Exemple 4 : Cet exemple décrit les instances Auto Scaling pour le groupe Auto Scaling spécifié.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1).Instances
```

Exemple 5 : Cet exemple décrit tous vos groupes Auto Scaling.

```
Get-ASAutoScalingGroup
```

Exemple 6 : Cet exemple décrit tous vos groupes Auto Scaling, par lots de 10.

```
$nextToken = $null
do {
  Get-ASAutoScalingGroup -NextToken $nextToken -MaxRecord 10
  $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

Exemple 7 : Cet LaunchTemplate exemple décrit le groupe Auto Scaling spécifié. Cet exemple suppose que les « Options d'achat d'instance » sont définies sur « Adhérer au modèle de lancement ». Si cette option est définie sur « Combiner les options d'achat et les types d'instances », elle est LaunchTemplate accessible à l'aide de « »MixedInstancesPolicy. LaunchTemplate« propriété.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-ag-1).LaunchTemplate
```

Sortie :

```
LaunchTemplateId     LaunchTemplateName   Version
----------------     ------------------   -------
lt-06095fd619cb40371 test-launch-template $Default
```

- Pour API plus de détails, consultez la section DescribeAutoScalingGroupsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ASAutoScalingInstance`

L'exemple de code suivant montre comment utiliser`Get-ASAutoScalingInstance`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie vos instances Auto Scaling. IDs

```
Get-ASAutoScalingInstance | format-table -property InstanceId
```

Sortie :

```
InstanceId
----------
i-12345678
i-87654321
i-abcd1234
```

Exemple 2 : Cet exemple décrit l'instance Auto Scaling spécifiée.

```
Get-ASAutoScalingInstance -InstanceId i-12345678
```

Sortie :

```
AutoScalingGroupName    : my-asg
AvailabilityZone        : us-west-2b
HealthStatus            : HEALTHY
InstanceId              : i-12345678
LaunchConfigurationName : my-lc
LifecycleState          : InService
```

Exemple 3 : Cet exemple décrit les deux instances Auto Scaling spécifiées.

```
Get-ASAutoScalingInstance -InstanceId @("i-12345678", "i-87654321")
```

Exemple 4 : Cet exemple décrit les instances Auto Scaling pour le groupe Auto Scaling spécifié.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg).Instances | Get-
ASAutoScalingInstance
```

Exemple 5 : Cet exemple décrit toutes vos instances d'Auto Scaling.

```
Get-ASAutoScalingInstance
```

Exemple 6 : Cet exemple décrit toutes vos instances d'Auto Scaling, par lots de 10.

```
$nextToken = $null
do {
  Get-ASAutoScalingInstance -NextToken $nextToken -MaxRecord 10
  $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Pour API plus de détails, consultez la section DescribeAutoScalingInstancesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ASAutoScalingNotificationType`

L'exemple de code suivant montre comment utiliser`Get-ASAutoScalingNotificationType`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie les types de notifications pris en charge par Auto Scaling.

```
Get-ASAutoScalingNotificationType
```

Sortie :

```
autoscaling:EC2_INSTANCE_LAUNCH
autoscaling:EC2_INSTANCE_LAUNCH_ERROR
autoscaling:EC2_INSTANCE_TERMINATE
autoscaling:EC2_INSTANCE_TERMINATE_ERROR
```

```
autoscaling:TEST_NOTIFICATION
```

- Pour API plus de détails, consultez la section DescribeAutoScalingNotificationTypesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ASLaunchConfiguration`

L'exemple de code suivant montre comment utiliser`Get-ASLaunchConfiguration`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie les noms de vos configurations de lancement.

```
Get-ASLaunchConfiguration | format-table -property LaunchConfigurationName
```

Sortie :

```
LaunchConfigurationName
-----------------------
my-lc-1
my-lc-2
my-lc-3
my-lc-4
my-lc-5
```

Exemple 2 : Cet exemple décrit la configuration de lancement spécifiée.

```
Get-ASLaunchConfiguration -LaunchConfigurationName my-lc-1
```

Sortie :

```
AssociatePublicIpAddress     : True
BlockDeviceMappings          : {/dev/xvda}
ClassicLinkVPCId             :
ClassicLinkVPCSecurityGroups : {}
CreatedTime                  : 12/12/2014 3:22:08 PM
EbsOptimized                 : False
IamInstanceProfile           :
ImageId                      : ami-043a5034
InstanceMonitoring           : Amazon.AutoScaling.Model.InstanceMonitoring
InstanceType                 : t2.micro
```

```
KernelId                    :
KeyName                     :
LaunchConfigurationARN      : arn:aws:autoscaling:us-
west-2:123456789012:launchConfiguration:7e5f31e4-693b-4604-9322-
                              e6f68d7fafad:launchConfigurationName/my-lc-1
LaunchConfigurationName     : my-lc-1
PlacementTenancy            :
RamdiskId                   :
SecurityGroups              : {sg-67ef0308}
SpotPrice                   :
UserData                    :
```

Exemple 3 : Cet exemple décrit les deux configurations de lancement spécifiées.

```
Get-ASLaunchConfiguration -LaunchConfigurationName @("my-lc-1", "my-lc-2")
```

Exemple 4 : Cet exemple décrit toutes vos configurations de lancement.

```
Get-ASLaunchConfiguration
```

Exemple 5 : Cet exemple décrit toutes vos configurations de lancement, par lots de 10.

```
$nextToken = $null
do {
  Get-ASLaunchConfiguration -NextToken $nextToken -MaxRecord 10
  $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Pour API plus de détails, consultez la section DescribeLaunchConfigurationsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ASLifecycleHook`

L'exemple de code suivant montre comment utiliser`Get-ASLifecycleHook`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit le hook de cycle de vie spécifié.

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName myLifecycleHook
```

Sortie :

```
AutoScalingGroupName  : my-asg
DefaultResult         : ABANDON
GlobalTimeout         : 172800
HeartbeatTimeout      : 3600
LifecycleHookName     : myLifecycleHook
LifecycleTransition   : auto-scaling:EC2_INSTANCE_LAUNCHING
NotificationMetadata  :
NotificationTargetARN : arn:aws:sns:us-west-2:123456789012:my-topic
RoleARN               : arn:aws:iam::123456789012:role/my-iam-role
```

Exemple 2 : Cet exemple décrit tous les hooks du cycle de vie pour le groupe Auto Scaling spécifié.

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg
```

Exemple 3 : Cet exemple décrit tous les hooks du cycle de vie pour tous vos groupes Auto Scaling.

```
Get-ASLifecycleHook
```

- Pour API plus de détails, consultez la section DescribeLifecycleHooksRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ASLifecycleHookType`

L'exemple de code suivant montre comment utiliser`Get-ASLifecycleHookType`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie les types de hooks de cycle de vie pris en charge par Auto Scaling.

```
Get-ASLifecycleHookType
```

Sortie :

```
autoscaling:EC2_INSTANCE_LAUNCHING
```

```
auto-scaling:EC2_INSTANCE_TERMINATING
```

- Pour API plus de détails, consultez la section DescribeLifecycleHookTypesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ASLoadBalancer`

L'exemple de code suivant montre comment utiliser`Get-ASLoadBalancer`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit les équilibreurs de charge pour le groupe Auto Scaling spécifié.

```
Get-ASLoadBalancer -AutoScalingGroupName my-asg
```

Sortie :

```
LoadBalancerName    State
----------------    -----
my-lb               Added
```

- Pour API plus de détails, consultez la section DescribeLoadBalancersRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ASMetricCollectionType`

L'exemple de code suivant montre comment utiliser`Get-ASMetricCollectionType`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie les types de collecte de métriques pris en charge par Auto Scaling.

```
(Get-ASMetricCollectionType).Metrics
```

Sortie :

```
Metric
------
```

```
GroupMinSize
GroupMaxSize
GroupDesiredCapacity
GroupInServiceInstances
GroupPendingInstances
GroupTerminatingInstances
GroupStandbyInstances
GroupTotalInstances
```

Exemple 2 : Cet exemple répertorie les granularités correspondantes.

```
(Get-ASMetricCollectionType).Granularities
```

Sortie :

```
Granularity
-----------
1Minute
```

- Pour API plus de détails, consultez la section DescribeMetricCollectionTypesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ASNotificationConfiguration`

L'exemple de code suivant montre comment utiliser`Get-ASNotificationConfiguration`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit les actions de notification associées au groupe Auto Scaling spécifié.

```
Get-ASNotificationConfiguration -AutoScalingGroupName my-asg | format-list
```

Sortie :

```
AutoScalingGroupName : my-asg
NotificationType     : auto-scaling:EC2_INSTANCE_LAUNCH
TopicARN             : arn:aws:sns:us-west-2:123456789012:my-topic

AutoScalingGroupName : my-asg
```

```
NotificationType     : auto-scaling:EC2_INSTANCE_TERMINATE
TopicARN             : arn:aws:sns:us-west-2:123456789012:my-topic
```

Exemple 2 : Cet exemple décrit les actions de notification associées à tous vos groupes Auto Scaling.

```
Get-ASNotificationConfiguration
```

- Pour API plus de détails, consultez la section DescribeNotificationConfigurationsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ASPolicy`

L'exemple de code suivant montre comment utiliser`Get-ASPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit toutes les politiques pour le groupe Auto Scaling spécifié.

```
Get-ASPolicy -AutoScalingGroupName my-asg
```

Sortie :

```
AdjustmentType          : ChangeInCapacity
Alarms                  : {}
AutoScalingGroupName    : my-asg
Cooldown                : 0
EstimatedInstanceWarmup : 0
MetricAggregationType   :
MinAdjustmentMagnitude  : 0
MinAdjustmentStep       : 0
PolicyARN               : arn:aws:auto-scaling:us-
west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-e1d769fc24ef
                          :autoScalingGroupName/my-asg:policyName/myScaleInPolicy
PolicyName              : myScaleInPolicy
PolicyType              : SimpleScaling
ScalingAdjustment       : -1
StepAdjustments         : {}
```

Exemple 2 : Cet exemple décrit les politiques spécifiées pour le groupe Auto Scaling spécifié.

```
Get-ASPolicy -AutoScalingGroupName my-asg -PolicyName @("myScaleOutPolicy",
 "myScaleInPolicy")
```

Exemple 3 : Cet exemple décrit toutes les politiques pour tous vos groupes Auto Scaling.

```
Get-ASPolicy
```

- Pour API plus de détails, consultez la section DescribePoliciesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ASScalingActivity`

L'exemple de code suivant montre comment utiliser`Get-ASScalingActivity`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit les activités de dimensionnement des six dernières semaines pour le groupe Auto Scaling spécifié.

```
Get-ASScalingActivity -AutoScalingGroupName my-asg
```

Sortie :

```
ActivityId           : 063308ae-aa22-4a9b-94f4-9fae4EXAMPLE
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:45:16Z a user request explicitly set group
 desired capacity changing the desired
                       capacity from 1 to 2.  At 2015-11-22T15:45:34Z an instance
 was started in response to a difference
                       between desired and actual capacity, increasing the capacity
 from 1 to 2.
Description          : Launching a new EC2 instance: i-26e715fc
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              : 11/22/2015 7:46:09 AM
Progress             : 100
StartTime            : 11/22/2015 7:45:35 AM
StatusCode           : Successful
StatusMessage        :
```

```
ActivityId           : ce719997-086d-4c73-a2f1-ab703EXAMPLE
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:57:53Z a user request created an
 AutoScalingGroup changing the desired capacity
                        from 0 to 1.  At 2015-11-20T22:57:58Z an instance was
 started in response to a difference betwe
                       en desired and actual capacity, increasing the capacity from
 0 to 1.
Description          : Launching a new EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              : 11/20/2015 2:58:32 PM
Progress             : 100
StartTime            : 11/20/2015 2:57:59 PM
StatusCode           : Successful
StatusMessage        :
```

Exemple 2 : Cet exemple décrit l'activité de dimensionnement spécifiée.

```
Get-ASScalingActivity -ActivityId "063308ae-aa22-4a9b-94f4-9fae4EXAMPLE"
```

Exemple 3 : Cet exemple décrit les activités de dimensionnement des six dernières semaines pour tous vos groupes Auto Scaling.

```
Get-ASScalingActivity
```

- Pour API plus de détails, consultez la section DescribeScalingActivitiesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ASScalingProcessType`

L'exemple de code suivant montre comment utiliser`Get-ASScalingProcessType`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie les types de processus pris en charge par Auto Scaling.

```
Get-ASScalingProcessType
```

Sortie :

```
ProcessName
-----------
AZRebalance
AddToLoadBalancer
AlarmNotification
HealthCheck
Launch
ReplaceUnhealthy
ScheduledActions
Terminate
```

- Pour API plus de détails, consultez la section DescribeScalingProcessTypesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ASScheduledAction`

L'exemple de code suivant montre comment utiliser`Get-ASScheduledAction`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit les actions de dimensionnement planifiées pour le groupe Auto Scaling spécifié.

```
Get-ASScheduledAction -AutoScalingGroupName my-asg
```

Sortie :

```
AutoScalingGroupName : my-asg
DesiredCapacity      : 10
EndTime              :
MaxSize              :
MinSize              :
Recurrence           :
ScheduledActionARN   : arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8a4c5f24-6ec6-4306-a2dd-f7
                       2c3af3a4d6:autoScalingGroupName/my-asg:scheduledActionName/
myScheduledAction
ScheduledActionName  : myScheduledAction
StartTime            : 11/30/2015 8:00:00 AM
Time                 : 11/30/2015 8:00:00 AM
```

Exemple 2 : Cet exemple décrit les actions de dimensionnement planifiées spécifiées.

```
Get-ASScheduledAction -ScheduledActionName @("myScheduledScaleOut",
 "myScheduledScaleIn")
```

Exemple 3 : Cet exemple décrit les actions de dimensionnement planifiées qui commencent à l'heure spécifiée.

```
Get-ASScheduledAction -StartTime "2015-12-01T08:00:00Z"
```

Exemple 4 : Cet exemple décrit les actions de dimensionnement planifiées qui se terminent à l'heure spécifiée.

```
Get-ASScheduledAction -EndTime "2015-12-30T08:00:00Z"
```

Exemple 5 : Cet exemple décrit les actions de dimensionnement planifiées pour tous vos groupes Auto Scaling.

```
Get-ASScheduledAction
```

- Pour API plus de détails, consultez la section DescribeScheduledActionsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ASTag`

L'exemple de code suivant montre comment utiliser`Get-ASTag`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit les balises dont la valeur clé est « myTag » ou « myTag 2 ». Les valeurs possibles pour le nom du filtre sont auto-scaling-group « », « key », « value » et « propagate-at-launch ». La syntaxe utilisée dans cet exemple nécessite PowerShell la version 3 ou ultérieure.

```
Get-ASTag -Filter @( @{ Name="key"; Values=@("myTag", "myTag2") } )
```

Sortie :

```
Key               : myTag2
PropagateAtLaunch : True
ResourceId        : my-asg
ResourceType      : auto-scaling-group
Value             : myTagValue2

Key               : myTag
PropagateAtLaunch : True
ResourceId        : my-asg
ResourceType      : auto-scaling-group
Value             : myTagValue
```

Exemple 2 : avec PowerShell la version 2, vous devez utiliser New-Object pour créer le filtre pour le paramètre Filter.

```
$keys = New-Object string[] 2
$keys[0] = "myTag"
$keys[1] = "myTag2"
$filter = New-Object Amazon.AutoScaling.Model.Filter
$filter.Name = "key"
$filter.Values = $keys
Get-ASTag -Filter @( $filter )
```

Exemple 3 : Cet exemple décrit toutes les balises de tous vos groupes Auto Scaling.

```
Get-ASTag
```

- Pour API plus de détails, consultez la section DescribeTagsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ASTerminationPolicyType`

L'exemple de code suivant montre comment utiliser`Get-ASTerminationPolicyType`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie les politiques de résiliation prises en charge par Auto Scaling.

```
Get-ASTerminationPolicyType
```

Sortie :

```
ClosestToNextInstanceHour
Default
NewestInstance
OldestInstance
OldestLaunchConfiguration
```

- Pour API plus de détails, consultez la section DescribeTerminationPolicyTypesRéférence des AWS Tools for PowerShell applets de commande.

### `Mount-ASInstance`

L'exemple de code suivant montre comment utiliser`Mount-ASInstance`.

Outils pour PowerShell

Exemple 1 : Cet exemple attache l'instance spécifiée au groupe Auto Scaling spécifié. Auto Scaling augmente automatiquement la capacité souhaitée du groupe Auto Scaling.

```
Mount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

- Pour API plus de détails, consultez la section AttachInstancesRéférence des AWS Tools for PowerShell applets de commande.

### `New-ASAutoScalingGroup`

L'exemple de code suivant montre comment utiliser`New-ASAutoScalingGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un groupe Auto Scaling avec le nom et les attributs spécifiés. La capacité souhaitée par défaut est la taille minimale. Par conséquent, ce groupe Auto Scaling lance deux instances, une dans chacune des deux zones de disponibilité spécifiées.

```
New-ASAutoScalingGroup -AutoScalingGroupName my-asg -LaunchConfigurationName my-lc -
MinSize 2 -MaxSize 6 -AvailabilityZone @("us-west-2a", "us-west-2b")
```

- Pour API plus de détails, consultez la section CreateAutoScalingGroupRéférence des AWS Tools for PowerShell applets de commande.

### `New-ASLaunchConfiguration`

L'exemple de code suivant montre comment utiliser`New-ASLaunchConfiguration`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une configuration de lancement nommée « my-lc ». Les EC2 instances lancées par les groupes Auto Scaling qui utilisent cette configuration de lancement utilisent le type d'instanceAMI, le groupe de sécurité et IAM le rôle spécifiés.

```
New-ASLaunchConfiguration -LaunchConfigurationName my-lc -InstanceType "m3.medium" -
ImageId "ami-12345678" -SecurityGroup "sg-12345678" -IamInstanceProfile "myIamRole"
```

- Pour API plus de détails, consultez la section CreateLaunchConfigurationRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-ASAutoScalingGroup`

L'exemple de code suivant montre comment utiliser`Remove-ASAutoScalingGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le groupe Auto Scaling spécifié s'il ne possède aucune instance en cours d'exécution. Vous êtes invité à confirmer avant que l'opération ne se poursuive.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASAutoScalingGroup (DeleteAutoScalingGroup)" on Target
 "my-asg".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemple 2 : Si vous spécifiez le paramètre Force, aucune confirmation ne vous est demandée avant le début de l'opération.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -Force
```

Exemple 3 : Cet exemple supprime le groupe Auto Scaling spécifié et met fin à toutes les instances en cours d'exécution qu'il contient.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -ForceDelete $true -Force
```

- Pour API plus de détails, consultez la section DeleteAutoScalingGroupRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-ASLaunchConfiguration`

L'exemple de code suivant montre comment utiliser`Remove-ASLaunchConfiguration`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la configuration de lancement spécifiée si elle n'est pas attachée à un groupe Auto Scaling. Vous êtes invité à confirmer avant que l'opération ne se poursuive.

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASLaunchConfiguration (DeleteLaunchConfiguration)" on
 Target "my-lc".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemple 2 : Si vous spécifiez le paramètre Force, aucune confirmation ne vous est demandée avant le début de l'opération.

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc -Force
```

- Pour API plus de détails, consultez la section DeleteLaunchConfigurationRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-ASLifecycleHook`

L'exemple de code suivant montre comment utiliser`Remove-ASLifecycleHook`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le hook de cycle de vie spécifié pour le groupe Auto Scaling spécifié. Vous êtes invité à confirmer avant que l'opération ne se poursuive.

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASLifecycleHook (DeleteLifecycleHook)" on Target
 "myLifecycleHook".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemple 2 : Si vous spécifiez le paramètre Force, aucune confirmation ne vous est demandée avant le début de l'opération.

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook -Force
```

- Pour API plus de détails, consultez la section DeleteLifecycleHookRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-ASNotificationConfiguration`

L'exemple de code suivant montre comment utiliser`Remove-ASNotificationConfiguration`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime l'action de notification spécifiée. Vous êtes invité à confirmer avant que l'opération ne se poursuive.

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN
 "arn:aws:sns:us-west-2:123456789012:my-topic"
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASNotificationConfiguration
 (DeleteNotificationConfiguration)" on Target
"arn:aws:sns:us-west-2:123456789012:my-topic".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemple 2 : Si vous spécifiez le paramètre Force, aucune confirmation ne vous est demandée avant le début de l'opération.

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN
 "arn:aws:sns:us-west-2:123456789012:my-topic" -Force
```

- Pour API plus de détails, consultez la section DeleteNotificationConfigurationRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-ASPolicy`

L'exemple de code suivant montre comment utiliser`Remove-ASPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la politique spécifiée pour le groupe Auto Scaling spécifié. Vous êtes invité à confirmer avant que l'opération ne se poursuive.

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASPolicy (DeletePolicy)" on Target "myScaleInPolicy".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemple 2 : Si vous spécifiez le paramètre Force, aucune confirmation ne vous est demandée avant le début de l'opération.

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy -Force
```

- Pour API plus de détails, consultez la section DeletePolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-ASScheduledAction`

L'exemple de code suivant montre comment utiliser`Remove-ASScheduledAction`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime l'action planifiée spécifiée pour le groupe Auto Scaling spécifié. Vous êtes invité à confirmer avant que l'opération ne se poursuive.

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction
 "myScheduledAction"
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASScheduledAction (DeleteScheduledAction)" on Target
 "myScheduledAction".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemple 2 : Si vous spécifiez le paramètre Force, aucune confirmation ne vous est demandée avant le début de l'opération.

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction
 "myScheduledAction" -Force
```

- Pour API plus de détails, consultez la section DeleteScheduledActionRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-ASTag`

L'exemple de code suivant montre comment utiliser`Remove-ASTag`.

Exemple 1 : Cet exemple supprime la balise spécifiée du groupe Auto Scaling spécifié. Vous êtes invité à confirmer avant que l'opération ne se poursuive. La syntaxe utilisée dans cet exemple nécessite PowerShell la version 3 ou ultérieure.

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag" } )
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ASTag (DeleteTags)" on target
 "Amazon.AutoScaling.Model.Tag".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemple 2 : Si vous spécifiez le paramètre Force, aucune confirmation ne vous est demandée avant le début de l'opération.

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag" } ) -Force
```

Exemple 3 : avec Powershell version 2, vous devez utiliser New-Object pour créer la balise pour le paramètre Tag.

```
$tag = New-Object Amazon.AutoScaling.Model.Tag
$tag.ResourceType = "auto-scaling-group"
$tag.ResourceId = "my-asg"
$tag.Key = "myTag"
Remove-ASTag -Tag $tag -Force
```

- Pour API plus de détails, consultez la section DeleteTagsRéférence des AWS Tools for PowerShell applets de commande.

### `Resume-ASProcess`

L'exemple de code suivant montre comment utiliser`Resume-ASProcess`.

Outils pour PowerShell

Exemple 1 : Cet exemple reprend le processus Auto Scaling spécifié pour le groupe Auto Scaling spécifié.

```
Resume-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"
```

Exemple 2 : Cet exemple reprend tous les processus Auto Scaling suspendus pour le groupe Auto Scaling spécifié.

```
Resume-ASProcess -AutoScalingGroupName my-asg
```

- Pour API plus de détails, consultez la section ResumeProcessesRéférence des AWS Tools for PowerShell applets de commande.

### `Set-ASDesiredCapacity`

L'exemple de code suivant montre comment utiliser`Set-ASDesiredCapacity`.

Outils pour PowerShell

Exemple 1 : Cet exemple définit la taille du groupe Auto Scaling spécifié.

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2
```

Exemple 2 : Cet exemple définit la taille du groupe Auto Scaling spécifié et attend la fin du temps de recharge avant de le redimensionner à la nouvelle taille.

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2 -HonorCooldown
 $true
```

- Pour API plus de détails, consultez la section SetDesiredCapacityRéférence des AWS Tools for PowerShell applets de commande.

### `Set-ASInstanceHealth`

L'exemple de code suivant montre comment utiliser`Set-ASInstanceHealth`.

Outils pour PowerShell

Exemple 1 : Cet exemple définit le statut de l'instance spécifiée sur « Non fonctionnelle », la mettant hors service. Auto Scaling met fin à l'instance et la remplace.

```
Set-ASInstanceHealth -HealthStatus Unhealthy -InstanceId i-93633f9b
```

Exemple 2 : Cet exemple définit le statut de l'instance spécifiée sur « Healthy », afin de la maintenir en service. Toute période de grâce relative au bilan de santé du groupe Auto Scaling n'est pas respectée.

```
Set-ASInstanceHealth -HealthStatus Healthy -InstanceId i-93633f9b -
ShouldRespectGracePeriod $false
```

- Pour API plus de détails, consultez la section SetInstanceHealthRéférence des AWS Tools for PowerShell applets de commande.

### `Set-ASInstanceProtection`

L'exemple de code suivant montre comment utiliser`Set-ASInstanceProtection`.

Outils pour PowerShell

Exemple 1 : Cet exemple active la protection de l'instance pour l'instance spécifiée.

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -
ProtectedFromScaleIn $true
```

Exemple 2 : Cet exemple désactive la protection de l'instance pour l'instance spécifiée.

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -
ProtectedFromScaleIn $false
```

- Pour API plus de détails, consultez la section SetInstanceProtectionRéférence des AWS Tools for PowerShell applets de commande.

### `Set-ASTag`

L'exemple de code suivant montre comment utiliser`Set-ASTag`.

Outils pour PowerShell

Exemple 1 : Cet exemple ajoute une seule balise au groupe Auto Scaling spécifié. La clé de balise est « myTag » et la valeur de la balise est « myTagValue ». Auto Scaling propage cette balise aux EC2 instances suivantes lancées par le groupe Auto Scaling. La syntaxe utilisée dans cet exemple nécessite PowerShell la version 3 ou ultérieure.

```
Set-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag"; Value="myTagValue"; PropagateAtLaunch=$true} )
```

Exemple 2 : avec PowerShell la version 2, vous devez utiliser New-Object pour créer la balise pour le paramètre Tag.

```
$tag = New-Object Amazon.AutoScaling.Model.Tag
$tag.ResourceType = "auto-scaling-group"
$tag.ResourceId = "my-asg"
$tag.Key = "myTag"
$tag.Value = "myTagValue"
$tag.PropagateAtLaunch = $true
Set-ASTag -Tag $tag
```

- Pour API plus de détails, consultez la section CreateOrUpdateTagsRéférence des AWS Tools for PowerShell applets de commande.

### `Start-ASPolicy`

L'exemple de code suivant montre comment utiliser`Start-ASPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple exécute la politique spécifiée pour le groupe Auto Scaling spécifié.

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy"
```

Exemple 2 : Cet exemple exécute la politique spécifiée pour le groupe Auto Scaling spécifié, après avoir attendu la fin du délai de recharge.

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy" -
HonorCooldown $true
```

- Pour API plus de détails, consultez la section ExecutePolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Stop-ASInstanceInAutoScalingGroup`

L'exemple de code suivant montre comment utiliser`Stop-ASInstanceInAutoScalingGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple met fin à l'instance spécifiée et diminue la capacité souhaitée de son groupe Auto Scaling afin qu'Auto Scaling ne lance pas d'instance de remplacement.

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $true
```

Sortie :

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5
AutoScalingGroupName :
Cause                : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out of
 service in response to a user
                       request, shrinking the capacity from 2 to 1.
Description          : Terminating EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 0
StartTime            : 11/22/2015 8:09:03 AM
StatusCode           : InProgress
StatusMessage        :
```

Exemple 2 : Cet exemple met fin à l'instance spécifiée sans diminuer la capacité souhaitée de son groupe Auto Scaling. Auto Scaling lance une instance de remplacement.

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $false
```

Sortie :

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5
```

```
AutoScalingGroupName :
Cause                : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out of
 service in response to a user
                        request.
Description          : Terminating EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 0
StartTime            : 11/22/2015 8:09:03 AM
StatusCode           : InProgress
StatusMessage        :
```

- Pour API plus de détails, consultez la section TerminateInstanceInAutoScalingGroupRéférence des AWS Tools for PowerShell applets de commande.

### `Suspend-ASProcess`

L'exemple de code suivant montre comment utiliser`Suspend-ASProcess`.

Outils pour PowerShell

Exemple 1 : Cet exemple suspend le processus Auto Scaling spécifié pour le groupe Auto Scaling spécifié.

```
Suspend-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"
```

Exemple 2 : Cet exemple suspend tous les processus Auto Scaling pour le groupe Auto Scaling spécifié.

```
Suspend-ASProcess -AutoScalingGroupName my-asg
```

- Pour API plus de détails, consultez la section SuspendProcessesRéférence des AWS Tools for PowerShell applets de commande.

### `Update-ASAutoScalingGroup`

L'exemple de code suivant montre comment utiliser`Update-ASAutoScalingGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple met à jour la taille minimale et maximale du groupe Auto Scaling spécifié.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -MaxSize 5 -MinSize 1
```

Exemple 2 : Cet exemple met à jour la période de recharge par défaut du groupe Auto Scaling spécifié.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -DefaultCooldown 10
```

Exemple 3 : Cet exemple met à jour les zones de disponibilité du groupe Auto Scaling spécifié.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -AvailabilityZone @("us-
west-2a", "us-west-2b")
```

Exemple 4 : Cet exemple met à jour le groupe Auto Scaling spécifié pour utiliser les contrôles de santé d'Elastic Load Balancing.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -HealthCheckType ELB -
HealthCheckGracePeriod 60
```

- Pour API plus de détails, consultez la section UpdateAutoScalingGroupRéférence des AWS Tools for PowerShell applets de commande.

### `Write-ASLifecycleActionHeartbeat`

L'exemple de code suivant montre comment utiliser`Write-ASLifecycleActionHeartbeat`.

Outils pour PowerShell

Exemple 1 : Cet exemple enregistre un battement de cœur pour l'action du cycle de vie spécifiée. Cela permet de maintenir l'instance en attente jusqu'à ce que vous ayez terminé l'action personnalisée.

```
Write-ASLifecycleActionHeartbeat -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook -LifecycleActionToken bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- Pour API plus de détails, consultez la section RecordLifecycleActionHeartbeatRéférence des AWS Tools for PowerShell applets de commande.

### `Write-ASLifecycleHook`

L'exemple de code suivant montre comment utiliser`Write-ASLifecycleHook`.

Outils pour PowerShell

Exemple 1 : Cet exemple ajoute le hook de cycle de vie spécifié au groupe Auto Scaling spécifié.

```
Write-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 "myLifecycleHook" -LifecycleTransition "autoscaling:EC2_INSTANCE_LAUNCHING" -
NotificationTargetARN "arn:aws:sns:us-west-2:123456789012:my-sns-topic" -RoleARN
 "arn:aws:iam::123456789012:role/my-iam-role"
```

- Pour API plus de détails, consultez la section PutLifecycleHookRéférence des AWS Tools for PowerShell applets de commande.

### `Write-ASNotificationConfiguration`

L'exemple de code suivant montre comment utiliser`Write-ASNotificationConfiguration`.

Outils pour PowerShell

Exemple 1 : Cet exemple configure le groupe Auto Scaling spécifié pour envoyer une notification à la SNS rubrique spécifiée lorsqu'il lance des EC2 instances.

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType
 "autoscaling:EC2_INSTANCE_LAUNCH" -TopicARN "arn:aws:sns:us-west-2:123456789012:my-
topic"
```

Exemple 2 : Cet exemple configure le groupe Auto Scaling spécifié pour envoyer une notification à la SNS rubrique spécifiée lorsqu'il lance ou met fin EC2 à des instances.

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType
 @("autoscaling:EC2_INSTANCE_LAUNCH", "autoscaling:EC2_INSTANCE_TERMINATE") -
TopicARN "arn:aws:sns:us-west-2:123456789012:my-topic"
```

- Pour API plus de détails, consultez la section PutNotificationConfigurationRéférence des AWS Tools for PowerShell applets de commande.

### `Write-ASScalingPolicy`

L'exemple de code suivant montre comment utiliser`Write-ASScalingPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple ajoute la politique spécifiée au groupe Auto Scaling spécifié. Le type de réglage spécifié détermine la manière d'interpréter le ScalingAdjustment paramètre. Avec « ChangeInCapacity », une valeur positive augmente la capacité du nombre d'instances spécifié et une valeur négative diminue la capacité du nombre d'instances spécifié.

```
Write-ASScalingPolicy -AutoScalingGroupName my-asg -AdjustmentType
 "ChangeInCapacity" -PolicyName "myScaleInPolicy" -ScalingAdjustment -1
```

Sortie :

```
arn:aws:autoscaling:us-west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-
e1d769fc24ef:autoScalingGroupName/my-asg
:policyName/myScaleInPolicy
```

- Pour API plus de détails, consultez la section PutScalingPolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Write-ASScheduledUpdateGroupAction`

L'exemple de code suivant montre comment utiliser`Write-ASScheduledUpdateGroupAction`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée ou met à jour une action planifiée unique pour modifier la capacité souhaitée à l'heure de début spécifiée.

```
Write-ASScheduledUpdateGroupAction -AutoScalingGroupName my-asg -ScheduledActionName
 "myScheduledAction" -StartTime "2015-12-01T00:00:00Z" -DesiredCapacity 10
```

- Pour API plus de détails, consultez la section PutScheduledUpdateGroupActionRéférence des AWS Tools for PowerShell applets de commande.

# AWS Budgets exemples utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS Tools for PowerShell with AWS Budgets.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques



## Actions

### `New-BGTBudget`

L'exemple de code suivant montre comment utiliser`New-BGTBudget`.

Outils pour PowerShell

Exemple 1 : crée un nouveau budget avec les contraintes budgétaires et temporelles spécifiées avec des notifications par e-mail.

```
$notification = @{
    NotificationType = "ACTUAL"
    ComparisonOperator = "GREATER_THAN"
    Threshold = 80
}

$addressObject = @{
    Address = @("user@domain.com")
    SubscriptionType = "EMAIL"
}

$subscriber = New-Object Amazon.Budgets.Model.NotificationWithSubscribers
$subscriber.Notification = $notification
$subscriber.Subscribers.Add($addressObject)
```

```
$startDate = [datetime]::new(2017,09,25)
$endDate = [datetime]::new(2017,10,25)

New-BGTBudget -Budget_BudgetName "Tester" -Budget_BudgetType COST -
CostTypes_IncludeTax $true -Budget_TimeUnit MONTHLY -BudgetLimit_Unit USD -
TimePeriod_Start $startDate -TimePeriod_End $endDate -AccountId 123456789012 -
BudgetLimit_Amount 200 -NotificationsWithSubscriber $subscriber
```

- Pour API plus de détails, consultez la section CreateBudgetRéférence des AWS Tools for PowerShell applets de commande.

# AWS Cloud9 exemples utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS Tools for PowerShell with AWS Cloud9.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `Get-C9EnvironmentData`

L'exemple de code suivant montre comment utiliser`Get-C9EnvironmentData`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir des informations sur les environnements de développement AWS Cloud9 spécifiés.

```
Get-C9EnvironmentData -EnvironmentId
 685f892f431b45c2b28cb69eadcdb0EX,1980b80e5f584920801c09086667f0EX
```

Sortie :

```
Arn         : arn:aws:cloud9:us-
east-1:123456789012:environment:685f892f431b45c2b28cb69eadcdb0EX
Description : Created from CodeStar.
Id          : 685f892f431b45c2b28cb69eadcdb0EX
Lifecycle   : Amazon.Cloud9.Model.EnvironmentLifecycle
Name        : my-demo-ec2-env
OwnerArn    : arn:aws:iam::123456789012:user/MyDemoUser
Type        : ec2

Arn         : arn:aws:cloud9:us-
east-1:123456789012:environment:1980b80e5f584920801c09086667f0EX
Description :
Id          : 1980b80e5f584920801c09086667f0EX
Lifecycle   : Amazon.Cloud9.Model.EnvironmentLifecycle
Name        : my-demo-ssh-env
OwnerArn    : arn:aws:iam::123456789012:user/MyDemoUser
Type        : ssh
```

Exemple 2 : Cet exemple permet d'obtenir des informations sur l'état du cycle de vie de l'environnement de développement AWS Cloud9 spécifié.

```
(Get-C9EnvironmentData -EnvironmentId 685f892f431b45c2b28cb69eadcdb0EX).Lifecycle
```

Sortie :

```
FailureResource Reason Status
--------------- ------ ------
                       CREATED
```

- Pour API plus de détails, consultez la section DescribeEnvironmentsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-C9EnvironmentList`

L'exemple de code suivant montre comment utiliser`Get-C9EnvironmentList`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir une liste des identifiants d'environnement de développement AWS Cloud9 disponibles.

```
Get-C9EnvironmentList
```

Sortie :

```
685f892f431b45c2b28cb69eadcdb0EX
1980b80e5f584920801c09086667f0EX
```

- Pour API plus de détails, consultez la section ListEnvironmentsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-C9EnvironmentMembershipList`

L'exemple de code suivant montre comment utiliser`Get-C9EnvironmentMembershipList`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir des informations sur les membres de l'environnement de développement AWS Cloud9 spécifié.

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

Sortie :

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-write
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX

EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

Exemple 2 : Cet exemple permet d'obtenir des informations sur le propriétaire de l'environnement de développement AWS Cloud9 spécifié.

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -
Permission owner
```

Sortie :

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

Exemple 3 : Cet exemple permet d'obtenir des informations sur le membre d'environnement spécifié pour plusieurs environnements de développement AWS Cloud9.

```
Get-C9EnvironmentMembershipList -UserArn arn:aws:iam::123456789012:user/MyDemoUser
```

Sortie :

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/17/2018 7:48:14 PM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX

EnvironmentId : 1980b80e5f584920801c09086667f0EX
LastAccess    : 1/16/2018 11:21:24 PM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

- Pour API plus de détails, consultez la section DescribeEnvironmentMembershipsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-C9EnvironmentStatus`

L'exemple de code suivant montre comment utiliser`Get-C9EnvironmentStatus`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir des informations d'état pour l'environnement de développement AWS Cloud9 spécifié.

```
Get-C9EnvironmentStatus -EnvironmentId 349c86d4579e4e7298d500ff57a6b2EX
```

Sortie :

```
Message                    Status
-------                    ------
Environment is ready to use ready
```

- Pour API plus de détails, consultez la section DescribeEnvironmentStatusRéférence des AWS Tools for PowerShell applets de commande.

### `New-C9EnvironmentEC2`

L'exemple de code suivant montre comment utiliser`New-C9EnvironmentEC2`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un environnement de développement AWS Cloud9 avec les paramètres spécifiés, lance une instance Amazon Elastic Compute Cloud EC2 (Amazon), puis se connecte de l'instance à l'environnement.

```
New-C9EnvironmentEC2 -Name my-demo-env -AutomaticStopTimeMinutes 60 -Description
 "My demonstration development environment." -InstanceType t2.micro -OwnerArn
 arn:aws:iam::123456789012:user/MyDemoUser -SubnetId subnet-d43a46EX
```

Sortie :

```
ffd88420d4824eeeaeaa8a04bfde8cEX
```

- Pour API plus de détails, reportez-vous à la section CreateEnvironmentEc2 du manuel AWS Tools for PowerShell Cmdlet Reference.

### `New-C9EnvironmentMembership`

L'exemple de code suivant montre comment utiliser`New-C9EnvironmentMembership`.

Outils pour PowerShell

Exemple 1 : Cet exemple ajoute le membre d'environnement spécifié à l'environnement de développement AWS Cloud9 spécifié.

```
New-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/AnotherDemoUser
 -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Permission read-write
```

Sortie :

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-write
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX
```

- Pour API plus de détails, consultez la section CreateEnvironmentMembershipRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-C9Environment`

L'exemple de code suivant montre comment utiliser`Remove-C9Environment`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime l'environnement de développement AWS Cloud9 spécifié. Si une EC2 instance Amazon est connectée à l'environnement, elle met également fin à l'instance.

```
Remove-C9Environment -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

- Pour API plus de détails, consultez la section DeleteEnvironmentRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-C9EnvironmentMembership`

L'exemple de code suivant montre comment utiliser`Remove-C9EnvironmentMembership`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le membre d'environnement spécifié de l'environnement de développement AWS Cloud9 spécifié.

```
Remove-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

- Pour API plus de détails, consultez la section DeleteEnvironmentMembershipRéférence des AWS Tools for PowerShell applets de commande.

### `Update-C9Environment`

L'exemple de code suivant montre comment utiliser`Update-C9Environment`.

Outils pour PowerShell

Exemple 1 : Cet exemple modifie les paramètres spécifiés de l'environnement de développement AWS Cloud9 existant spécifié.

```
Update-C9Environment -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Description
 "My changed demonstration development environment." -Name my-changed-demo-env
```

- Pour API plus de détails, consultez la section UpdateEnvironmentRéférence des AWS Tools for PowerShell applets de commande.

### `Update-C9EnvironmentMembership`

L'exemple de code suivant montre comment utiliser`Update-C9EnvironmentMembership`.

Outils pour PowerShell

Exemple 1 : Cet exemple modifie les paramètres du membre d'environnement existant spécifié pour l'environnement de développement AWS Cloud9 spécifié.

```
Update-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Permission read-
only
```

Sortie :

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
```

```
Permissions   : read-only
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX
```

- Pour API plus de détails, consultez la section UpdateEnvironmentMembershipRéférence des AWS Tools for PowerShell applets de commande.

# AWS CloudFormation exemples utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS Tools for PowerShell with AWS CloudFormation.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `Get-CFNStack`

L'exemple de code suivant montre comment utiliser`Get-CFNStack`.

Outils pour PowerShell

Exemple 1 : renvoie une collection d'instances Stack décrivant toutes les piles de l'utilisateur.

```
Get-CFNStack
```

Exemple 2 : Renvoie une instance de Stack décrivant la pile spécifiée

```
Get-CFNStack -StackName "myStack"
```

Exemple 3 : renvoie une collection d'instances Stack décrivant toutes les piles de l'utilisateur à l'aide de la pagination manuelle. Le jeton de départ pour la page suivante est récupéré après chaque appel avec $null indiquant qu'il ne reste plus de détails à récupérer.

```
$nextToken = $null
do {
    Get-CFNStack -NextToken $nextToken
    $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Pour API plus de détails, consultez la section DescribeStacksRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFNStackEvent`

L'exemple de code suivant montre comment utiliser`Get-CFNStackEvent`.

Outils pour PowerShell

Exemple 1 : renvoie tous les événements liés à la pile pour la pile spécifiée.

```
Get-CFNStackEvent -StackName "myStack"
```

Exemple 2 : renvoie tous les événements liés à la pile pour la pile spécifiée en utilisant une pagination manuelle à partir du jeton spécifié. Le jeton de départ pour la page suivante est récupéré après chaque appel avec $null indiquant qu'il ne reste plus aucun événement à récupérer.

```
$nextToken = $null
do {
    Get-CFNStack -StackName "myStack" -NextToken $nextToken
    $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Pour API plus de détails, consultez la section DescribeStackEventsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFNStackResource`

L'exemple de code suivant montre comment utiliser`Get-CFNStackResource`.

Outils pour PowerShell

Exemple 1 : renvoie la description d'une ressource identifiée dans le modèle associé à la pile spécifiée par l'ID logique yDBInstance « M ».

```
Get-CFNStackResource -StackName "myStack" -LogicalResourceId "MyDBInstance"
```

- Pour API plus de détails, consultez la section DescribeStackResourceRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFNStackResourceList`

L'exemple de code suivant montre comment utiliser`Get-CFNStackResourceList`.

Outils pour PowerShell

Exemple 1 : renvoie les descriptions de AWS ressources pour un maximum de 100 ressources associées à la pile spécifiée. Pour obtenir des informations détaillées sur toutes les ressources associées à une pile, utilisez le Get-CFNStackResourceSummary, qui prend également en charge la pagination manuelle des résultats.

```
Get-CFNStackResourceList -StackName "myStack"
```

Exemple 2 : renvoie la description de l'EC2instance Amazon identifiée dans le modèle associé à la pile spécifiée par l'ID logique « Ec2Instance ».

```
Get-CFNStackResourceList -StackName "myStack" -LogicalResourceId "Ec2Instance"
```

Exemple 3 : renvoie la description d'un maximum de 100 ressources associées à la pile contenant une EC2 instance Amazon identifiée par l'ID d'instance « i-123456 ». Pour obtenir des informations détaillées sur toutes les ressources associées à une pile, utilisez le Get-CFNStackResourceSummary, qui prend également en charge la pagination manuelle des résultats.

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456"
```

Exemple 4 : renvoie la description de l'EC2instance Amazon identifiée par l'ID logique « Ec2Instance » dans le modèle d'une pile. La pile est identifiée à l'aide de l'ID de ressource physique d'une ressource qu'elle contient, dans ce cas également une EC2 instance Amazon

avec l'ID d'instance « i-123456 ». Une ressource physique différente peut également être utilisée pour identifier la pile en fonction du contenu du modèle, par exemple un compartiment Amazon S3.

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456" -LogicalResourceId
 "Ec2Instance"
```

- Pour API plus de détails, consultez la section DescribeStackResourcesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFNStackResourceSummary`

L'exemple de code suivant montre comment utiliser`Get-CFNStackResourceSummary`.

Outils pour PowerShell

Exemple 1 : renvoie les descriptions de toutes les ressources associées à la pile spécifiée.

```
Get-CFNStackResourceSummary -StackName "myStack"
```

Exemple 2 : renvoie les descriptions de toutes les ressources associées à la pile spécifiée en utilisant la pagination manuelle des résultats. Le jeton de départ pour la page suivante est récupéré après chaque appel avec $null indiquant qu'il ne reste plus de détails à récupérer.

```
$nextToken = $null
do {
  Get-CFNStackResourceSummary -StackName "myStack" -NextToken $nextToken
  $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Pour API plus de détails, consultez la section ListStackResourcesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFNStackSummary`

L'exemple de code suivant montre comment utiliser`Get-CFNStackSummary`.

Outils pour PowerShell

Exemple 1 : renvoie des informations récapitulatives pour toutes les piles.

```
Get-CFNStackSummary
```

Exemple 2 : renvoie des informations récapitulatives pour toutes les piles en cours de création.

```
Get-CFNStackSummary -StackStatusFilter "CREATE_IN_PROGRESS"
```

Exemple 3 : renvoie des informations récapitulatives pour toutes les piles en cours de création ou de mise à jour.

```
Get-CFNStackSummary -StackStatusFilter @("CREATE_IN_PROGRESS", "UPDATE_IN_PROGRESS")
```

Exemple 4 : renvoie des informations récapitulatives pour toutes les piles en cours de création ou de mise à jour à l'aide de la pagination manuelle des résultats. Le jeton de départ pour la page suivante est récupéré après chaque appel avec $null indiquant qu'il ne reste plus de détails à récupérer.

```
$nextToken = $null
do {
  Get-CFNStackSummary -StackStatusFilter @("CREATE_IN_PROGRESS",
 "UPDATE_IN_PROGRESS") -NextToken $nextToken
  $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Pour API plus de détails, consultez la section ListStacksRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFNTemplate`

L'exemple de code suivant montre comment utiliser`Get-CFNTemplate`.

Outils pour PowerShell

Exemple 1 : renvoie le modèle associé à la pile spécifiée.

```
Get-CFNTemplate -StackName "myStack"
```

- Pour API plus de détails, consultez la section GetTemplateRéférence des AWS Tools for PowerShell applets de commande.

### `Measure-CFNTemplateCost`

L'exemple de code suivant montre comment utiliser`Measure-CFNTemplateCost`.

Outils pour PowerShell

Exemple 1 : renvoie une calculatrice mensuelle AWS simple URL avec une chaîne de requête qui décrit les ressources nécessaires pour exécuter le modèle. Le modèle est obtenu à partir de l'Amazon S3 spécifié URL et le paramètre de personnalisation unique est appliqué. Le paramètre peut également être spécifié en utilisant « Key » et « Value » au lieu de ParameterKey « » et « ParameterValue ».

```
Measure-CFNTemplateCost -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                        -Region us-west-1 `
                        -Parameter @{ ParameterKey="KeyName";
 ParameterValue="myKeyPairName" }
```

Exemple 2 : renvoie une calculatrice mensuelle AWS simple URL avec une chaîne de requête qui décrit les ressources nécessaires pour exécuter le modèle. Le modèle est analysé à partir du contenu fourni et des paramètres de personnalisation appliqués (cet exemple suppose que le contenu du modèle aurait déclaré deux paramètres, « KeyName » et « InstanceType »). Les paramètres de personnalisation peuvent également être spécifiés en utilisant « Key » et « Value » au lieu de ParameterKey « » et « ParameterValue ».

```
Measure-CFNTemplateCost -TemplateBody "{TEMPLATE CONTENT HERE}" `
                        -Parameter @( @{ ParameterKey="KeyName";
 ParameterValue="myKeyPairName" },`
                                     @{ ParameterKey="InstanceType";
 ParameterValue="m1.large" })
```

Exemple 3 : Utilise New-Object pour créer l'ensemble des paramètres du modèle et renvoie un calculateur mensuel AWS simple URL avec une chaîne de requête qui décrit les ressources requises pour exécuter le modèle. Le modèle est analysé à partir du contenu fourni, avec des paramètres de personnalisation (cet exemple suppose que le contenu du modèle aurait déclaré deux paramètres, « KeyName » et « InstanceType »).

```
$p1 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p1.ParameterKey = "KeyName"
```

```
$p1.ParameterValue = "myKeyPairName"

$p2 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p2.ParameterKey = "InstanceType"
$p2.ParameterValue = "m1.large"

Measure-CFNTemplateCost -TemplateBody "{TEMPLATE CONTENT HERE}" -Parameter @( $p1,
 $p2 )
```

- Pour API plus de détails, consultez la section EstimateTemplateCostRéférence des AWS Tools for PowerShell applets de commande.

### `New-CFNStack`

L'exemple de code suivant montre comment utiliser`New-CFNStack`.

Outils pour PowerShell

Exemple 1 : crée une nouvelle pile portant le nom spécifié. Le modèle est analysé à partir du contenu fourni avec des paramètres de personnalisation (PK1« » et « PK2 » représentent les noms des paramètres déclarés dans le contenu du modèle, « PV1 » et « PV2 » représentent les valeurs de ces paramètres. Les paramètres de personnalisation peuvent également être spécifiés en utilisant « Key » et « Value » au lieu de ParameterKey « » et « ParameterValue ». Si la création de la pile échoue, elle ne sera pas annulée.

```
New-CFNStack -StackName "myStack" `
             -TemplateBody "{TEMPLATE CONTENT HERE}" `
             -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" }) `
             -DisableRollback $true
```

Exemple 2 : crée une nouvelle pile portant le nom spécifié. Le modèle est analysé à partir du contenu fourni avec des paramètres de personnalisation (PK1« » et « PK2 » représentent les noms des paramètres déclarés dans le contenu du modèle, « PV1 » et « PV2 » représentent les valeurs de ces paramètres. Les paramètres de personnalisation peuvent également être spécifiés en utilisant « Key » et « Value » au lieu de ParameterKey « » et « ParameterValue ». Si la création de la pile échoue, elle sera annulée.

```
$p1 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p1.ParameterKey = "PK1"
```

```
$p1.ParameterValue = "PV1"

$p2 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p2.ParameterKey = "PK2"
$p2.ParameterValue = "PV2"

New-CFNStack -StackName "myStack" `
             -TemplateBody "{TEMPLATE CONTENT HERE}" `
             -Parameter @( $p1, $p2 ) `
             -OnFailure "ROLLBACK"
```

Exemple 3 : Crée une nouvelle pile portant le nom spécifié. Le modèle est obtenu auprès d'Amazon S3 URL avec des paramètres de personnalisation (PK1« » représente le nom d'un paramètre déclaré dans le contenu du modèle, PV1 « » représente la valeur du paramètre. Les paramètres de personnalisation peuvent également être spécifiés en utilisant « Key » et « Value » au lieu de ParameterKey « » et « ParameterValue ». Si la création de la pile échoue, elle sera annulée (comme si vous spécifiez - DisableRollback $false).

```
New-CFNStack -StackName "myStack" `
             -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
             -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Exemple 4 : Crée une nouvelle pile portant le nom spécifié. Le modèle est obtenu auprès d'Amazon S3 URL avec des paramètres de personnalisation (PK1« » représente le nom d'un paramètre déclaré dans le contenu du modèle, PV1 « » représente la valeur du paramètre. Les paramètres de personnalisation peuvent également être spécifiés en utilisant « Key » et « Value » au lieu de ParameterKey « » et « ParameterValue ». Si la création de la pile échoue, elle sera annulée (comme si vous spécifiez - DisableRollback $false). La notification spécifiée AENs recevra les événements publiés liés à la pile.

```
New-CFNStack -StackName "myStack" `
             -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
             -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" } `
             -NotificationARN @( "arn1", "arn2" )
```

- Pour API plus de détails, consultez la section CreateStackRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-CFNStack`

L'exemple de code suivant montre comment utiliser`Remove-CFNStack`.

Outils pour PowerShell

Exemple 1 : Supprime la pile spécifiée.

```
Remove-CFNStack -StackName "myStack"
```

- Pour API plus de détails, consultez la section DeleteStackRéférence des AWS Tools for PowerShell applets de commande.

### `Resume-CFNUpdateRollback`

L'exemple de code suivant montre comment utiliser`Resume-CFNUpdateRollback`.

Outils pour PowerShell

Exemple 1 : Poursuit l'annulation de la pile nommée, qui doit être dans l'état « UPDATE _ ROLLBACK _ FAILED ». Si la restauration continue est réussie, la pile entrera dans l'état « UPDATE _ ROLLBACK _ COMPLETE ».

```
Resume-CFNUpdateRollback -StackName "myStack"
```

- Pour API plus de détails, consultez la section ContinueUpdateRollbackRéférence des AWS Tools for PowerShell applets de commande.

### `Stop-CFNUpdateStack`

L'exemple de code suivant montre comment utiliser`Stop-CFNUpdateStack`.

Outils pour PowerShell

Exemple 1 : annule une mise à jour sur la pile spécifiée.

```
Stop-CFNUpdateStack -StackName "myStack"
```

- Pour API plus de détails, consultez la section CancelUpdateStackRéférence des AWS Tools for PowerShell applets de commande.

### `Test-CFNStack`

L'exemple de code suivant montre comment utiliser`Test-CFNStack`.

Outils pour PowerShell

Exemple 1 : teste si la pile a atteint l'un des états UPDATE _ ROLLBACK _COMPLETE, CREATE _COMPLETE, ROLLBACK _ COMPLETE ou UPDATE _COMPLETE.

```
Test-CFNStack -StackName MyStack
```

Sortie :

```
False
```

Exemple 2 : teste si la pile a atteint le statut UPDATE _ COMPLETE ou UPDATE _ ROLLBACK _COMPLETE.

```
Test-CFNStack -StackName MyStack -Status UPDATE_COMPLETE,UPDATE_ROLLBACK_COMPLETE
```

Sortie :

```
True
```

- Pour API plus de détails, consultez la section Test- CFNStack in AWS Tools for PowerShell Cmdlet Reference.

### `Test-CFNTemplate`

L'exemple de code suivant montre comment utiliser`Test-CFNTemplate`.

Outils pour PowerShell

Exemple 1 : valide le contenu du modèle spécifié. La sortie détaille les fonctionnalités, la description et les paramètres du modèle.

```
Test-CFNTemplate -TemplateBody "{TEMPLATE CONTENT HERE}"
```

Exemple 2 : valide le modèle spécifié accessible via un Amazon S3URL. La sortie détaille les fonctionnalités, la description et les paramètres du modèle.

```
Test-CFNTemplate -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template
```

- Pour API plus de détails, consultez la section ValidateTemplateRéférence des AWS Tools for PowerShell applets de commande.

### `Update-CFNStack`

L'exemple de code suivant montre comment utiliser`Update-CFNStack`.

Outils pour PowerShell

Exemple 1 : met à jour la pile « myStack » avec le modèle et les paramètres de personnalisation spécifiés. « PK1 » représente le nom d'un paramètre déclaré dans le modèle et « PV1 » représente sa valeur. Les paramètres de personnalisation peuvent également être spécifiés en utilisant « Key » et « Value » au lieu de ParameterKey « » et « ParameterValue ».

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Exemple 2 : met à jour la pile « myStack » avec le modèle et les paramètres de personnalisation spécifiés. « PK1 » et « PK2 » représentent les noms des paramètres déclarés dans le modèle, « PV1 » et « PV2 » représentent les valeurs demandées. Les paramètres de personnalisation peuvent également être spécifiés en utilisant « Key » et « Value » au lieu de ParameterKey « » et « ParameterValue ».

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

Exemple 3 : met à jour la pile « myStack » avec le modèle et les paramètres de personnalisation spécifiés. « PK1 » représente le nom d'un paramètre déclaré dans le modèle et « PV2 » représente sa valeur. Les paramètres de personnalisation peuvent également être spécifiés en utilisant « Key » et « Value » au lieu de ParameterKey « » et « ParameterValue ».

```
Update-CFNStack -StackName "myStack" -TemplateBody "{Template Content Here}" -
Parameters @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Exemple 4 : met à jour la pile « myStack » avec le modèle spécifié, obtenu auprès d'Amazon S3, et les paramètres de personnalisation. « PK1 » et « PK2 » représentent les noms des paramètres déclarés dans le modèle, « PV1 » et « PV2 » représentent les valeurs demandées. Les paramètres de personnalisation peuvent également être spécifiés en utilisant « Key » et « Value » au lieu de ParameterKey « » et « ParameterValue ».

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

Exemple 5 : met à jour la pile myStack « », censée contenir des IAM ressources dans cet exemple, avec le modèle spécifié, obtenu auprès d'Amazon S3, et les paramètres de personnalisation. « PK1 » et « PK2 » représentent les noms des paramètres déclarés dans le modèle, « PV1 » et « PV2 » représentent les valeurs demandées. Les paramètres de personnalisation peuvent également être spécifiés en utilisant « Key » et « Value » au lieu de ParameterKey « » et « ParameterValue ». Les piles contenant des IAM ressources nécessitent que vous spécifiiez le paramètre -Capabilities « CAPABILITY _ IAM », sinon la mise à jour échouera avec une erreur « InsufficientCapabilities ».

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } ) `
                -Capabilities "CAPABILITY_IAM"
```

- Pour API plus de détails, consultez la section UpdateStackRéférence des AWS Tools for PowerShell applets de commande.

### `Wait-CFNStack`

L'exemple de code suivant montre comment utiliser`Wait-CFNStack`.

Outils pour PowerShell

Exemple 1 : teste si la pile a atteint l'un des états UPDATE _ ROLLBACK _COMPLETE, CREATE _COMPLETE, ROLLBACK _ COMPLETE ou UPDATE _COMPLETE. Si la pile n'est pas dans

l'un des états, la commande est mise en veille pendant deux secondes avant de tester à nouveau l'état. Ceci est répété jusqu'à ce que la pile atteigne l'un des états demandés ou que le délai d'expiration par défaut de 60 secondes s'écoule. Si le délai d'expiration est dépassé, une exception est déclenchée. Si la pile atteint l'un des états demandés dans le délai imparti, elle est renvoyée dans le pipeline.

```
$stack = Wait-CFNStack -StackName MyStack
```

Exemple 2 : Cet exemple attend au total 5 minutes (300 secondes) pour que la pile atteigne l'un des états spécifiés. Dans cet exemple, l'état est atteint avant le délai d'expiration et l'objet de la pile est donc renvoyé dans le pipeline.

```
Wait-CFNStack -StackName MyStack -Timeout 300 -Status
 CREATE_COMPLETE,ROLLBACK_COMPLETE
```

Sortie :

```
Capabilities      : {CAPABILITY_IAM}
ChangeSetId       :
CreationTime      : 6/1/2017 9:29:33 AM
Description       : AWS CloudFormation Sample Template
 ec2_instance_with_instance_profile: Create an EC2 instance with an associated
 instance profile. **WARNING** This template creates one or more Amazon EC2
                   instances and an Amazon SQS queue. You will be billed for the
 AWS resources used if you create a stack from this template.
DisableRollback   : False
LastUpdatedTime   : 1/1/0001 12:00:00 AM
NotificationARNs  : {}
Outputs           : {}
Parameters        : {}
RoleARN           :
StackId           : arn:aws:cloudformation:us-west-2:123456789012:stack/
MyStack/7ea87b50-46e7-11e7-9c9b-503a90a9c4d1
StackName         : MyStack
StackStatus       : CREATE_COMPLETE
StackStatusReason :
Tags              : {}
TimeoutInMinutes  : 0
```

Exemple 3 : Cet exemple montre le résultat d'erreur lorsqu'une pile n'atteint pas l'un des états demandés dans le délai imparti (dans ce cas, le délai par défaut de 60 secondes).

```
Wait-CFNStack -StackName MyStack -Status CREATE_COMPLETE,ROLLBACK_COMPLETE
```

Sortie :

```
Wait-CFNStack : Timed out after 60 seconds waiting for CloudFormation
 stack MyStack in region us-west-2 to reach one of state(s):
 UPDATE_ROLLBACK_COMPLETE,CREATE_COMPLETE,ROLLBACK_COMPLETE,UPDATE_COMPLETE
At line:1 char:1
+ Wait-CFNStack -StackName MyStack -State CREATE_COMPLETE,ROLLBACK_COMPLETE
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation:
 (Amazon.PowerShe...tCFNStackCmdlet:WaitCFNStackCmdlet) [Wait-CFNStack],
 InvalidOperationException
    + FullyQualifiedErrorId :
 InvalidOperationException,Amazon.PowerShell.Cmdlets.CFN.WaitCFNStackCmdlet
```

- Pour API plus de détails, consultez la section Référence des AWS Tools for PowerShell applets de commande Wait- CFNStack in.

# CloudFront exemples utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS Tools for PowerShell with CloudFront.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `Get-CFCloudFrontOriginAccessIdentity`

L'exemple de code suivant montre comment utiliser`Get-CFCloudFrontOriginAccessIdentity`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie une identité d'accès Amazon CloudFront Origin spécifique, spécifiée par le paramètre -Id. Bien que le paramètre -Id ne soit pas obligatoire, aucun résultat n'est renvoyé si vous ne le spécifiez pas.

```
Get-CFCloudFrontOriginAccessIdentity -Id E3XXXXXXXXXXRT
```

Sortie :

```
    CloudFrontOriginAccessIdentityConfig    Id
S3CanonicalUserId
    ------------------------------------    --
-----------------
    Amazon.CloudFront.Model.CloudFrontOr... E3XXXXXXXXXXRT
4b6e...
```

- Pour API plus de détails, consultez la section GetCloudFrontOriginAccessIdentityRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFCloudFrontOriginAccessIdentityConfig`

L'exemple de code suivant montre comment utiliser`Get-CFCloudFrontOriginAccessIdentityConfig`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie des informations de configuration concernant une seule identité CloudFront d'accès à Amazon Origin, spécifiée par le paramètre -Id. Des erreurs se produisent si aucun paramètre -Id n'est spécifié.

```
Get-CFCloudFrontOriginAccessIdentityConfig -Id E3XXXXXXXXXXRT
```

Sortie :

```
    CallerReference                                         Comment
    ---------------                                         -------
    mycallerreference: 2/1/2011 1:16:32 PM                  Caller reference:
2/1/2011 1:16:32 PM
```

- Pour API plus de détails, consultez la section GetCloudFrontOriginAccessIdentityConfigRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFCloudFrontOriginAccessIdentityList`

L'exemple de code suivant montre comment utiliser`Get-CFCloudFrontOriginAccessIdentityList`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie une liste des identités CloudFront d'accès d'origine Amazon. Comme le MaxItem paramètre - spécifie une valeur de 2, les résultats incluent deux identités.

```
Get-CFCloudFrontOriginAccessIdentityList -MaxItem 2
```

Sortie :

```
IsTruncated : True
Items       : {E326XXXXXXXXXT, E1YWXXXXXXX9B}
Marker      :
MaxItems    : 2
NextMarker  : E1YXXXXXXXXX9B
Quantity    : 2
```

- Pour API plus de détails, consultez la section ListCloudFrontOriginAccessIdentitiesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFDistribution`

L'exemple de code suivant montre comment utiliser`Get-CFDistribution`.

Outils pour PowerShell

Exemple 1 : récupère les informations relatives à une distribution spécifique.

```
Get-CFDistribution -Id EXAMPLE0000ID
```

- Pour API plus de détails, consultez la section GetDistributionRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFDistributionConfig`

L'exemple de code suivant montre comment utiliser`Get-CFDistributionConfig`.

Outils pour PowerShell

Exemple 1 : récupère la configuration d'une distribution spécifique.

```
Get-CFDistributionConfig -Id EXAMPLE0000ID
```

- Pour API plus de détails, consultez la section GetDistributionConfigRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFDistributionList`

L'exemple de code suivant montre comment utiliser`Get-CFDistributionList`.

Outils pour PowerShell

Exemple 1 : renvoie les distributions.

```
Get-CFDistributionList
```

- Pour API plus de détails, consultez la section ListDistributionsRéférence des AWS Tools for PowerShell applets de commande.

### `New-CFDistribution`

L'exemple de code suivant montre comment utiliser`New-CFDistribution`.

Outils pour PowerShell

Exemple 1 : crée une CloudFront distribution de base, configurée avec la journalisation et la mise en cache.

```
$origin = New-Object Amazon.CloudFront.Model.Origin
$origin.DomainName = "amzn-s3-demo-bucket.s3.amazonaws.com"
$origin.Id = "UniqueOrigin1"
$origin.S3OriginConfig = New-Object Amazon.CloudFront.Model.S3OriginConfig
$origin.S3OriginConfig.OriginAccessIdentity = ""
New-CFDistribution `
      -DistributionConfig_Enabled $true `
```

```
    -DistributionConfig_Comment "Test distribution" `
    -Origins_Item $origin `
    -Origins_Quantity 1 `
    -Logging_Enabled $true `
    -Logging_IncludeCookie $true `
    -Logging_Bucket amzn-s3-demo-logging-bucket.s3.amazonaws.com `
    -Logging_Prefix "help/" `
    -DistributionConfig_CallerReference Client1 `
    -DistributionConfig_DefaultRootObject index.html `
    -DefaultCacheBehavior_TargetOriginId $origin.Id `
    -ForwardedValues_QueryString $true `
    -Cookies_Forward all `
    -WhitelistedNames_Quantity 0 `
    -TrustedSigners_Enabled $false `
    -TrustedSigners_Quantity 0 `
    -DefaultCacheBehavior_ViewerProtocolPolicy allow-all `
    -DefaultCacheBehavior_MinTTL 1000 `
    -DistributionConfig_PriceClass "PriceClass_All" `
    -CacheBehaviors_Quantity 0 `
    -Aliases_Quantity 0
```

- Pour API plus de détails, consultez la section CreateDistributionRéférence des AWS Tools for PowerShell applets de commande.

### `New-CFInvalidation`

L'exemple de code suivant montre comment utiliser`New-CFInvalidation`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une nouvelle invalidation sur une distribution dont l'EXAMPLENSTXAXEID est. CallerReference Il s'agit d'un identifiant unique choisi par l'utilisateur ; dans ce cas, un horodatage représentant le 15 mai 2019 à 9 h 00 est utilisé. La variable $Paths stocke trois chemins d'accès aux fichiers image et multimédia que l'utilisateur ne souhaite pas inclure dans le cache de la distribution. La valeur du paramètre -Paths_Quantity est le nombre total de chemins spécifiés dans le paramètre -Paths_Item.

```
$Paths = "/images/*.gif", "/images/image1.jpg", "/videos/*.mp4"
New-CFInvalidation -DistributionId "EXAMPLENSTXAXE" -
InvalidationBatch_CallerReference 20190515090000 -Paths_Item $Paths -Paths_Quantity
 3
```

Sortie :

```
Invalidation                          Location

------------                          --------

Amazon.CloudFront.Model.Invalidation https://cloudfront.amazonaws.com/2018-11-05/
distribution/EXAMPLENSTXAXE/invalidation/EXAMPLE8NOK9H
```

- Pour API plus de détails, consultez la section CreateInvalidationRéférence des AWS Tools for PowerShell applets de commande.

### `New-CFSignedCookie`

L'exemple de code suivant montre comment utiliser`New-CFSignedCookie`.

Outils pour PowerShell

Exemple 1 : crée un cookie signé sur la ressource spécifiée à l'aide d'une politique prédéfinie. Le cookie sera valide pendant un an.

```
$params = @{
 "ResourceUri"="http://xyz.cloudfront.net/image1.jpeg"
 "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
 "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
 "ExpiresOn"=(Get-Date).AddYears(1)
}
New-CFSignedCookie @params
```

Sortie :

```
Expires
-------
[CloudFront-Expires, 1472227284]
```

Exemple 2 : crée un cookie signé pour les ressources spécifiées à l'aide d'une politique personnalisée. Le cookie sera valide dans les 24 heures et expirera une semaine plus tard.

```
$start = (Get-Date).AddHours(24)
$params = @{
```

```
 "ResourceUri"="http://xyz.cloudfront.net/content/*.jpeg"
 "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
 "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
 "ExpiresOn"=$start.AddDays(7)
    "ActiveFrom"=$start
}

New-CFSignedCookie @params
```

Sortie :

```
Policy
------
[CloudFront-Policy, eyJTd...wIjo...
```

Exemple 3 : crée un cookie signé pour les ressources spécifiées à l'aide d'une politique personnalisée. Le cookie sera valide dans les 24 heures et expirera une semaine plus tard. L'accès aux ressources est limité à la plage d'adresses IP spécifiée.

```
$start = (Get-Date).AddHours(24)
$params = @{
 "ResourceUri"="http://xyz.cloudfront.net/content/*.jpeg"
 "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
 "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
 "ExpiresOn"=$start.AddDays(7)
    "ActiveFrom"=$start
 "IpRange"="192.0.2.0/24"
}

New-CFSignedCookie @params
```

Sortie :

```
Policy
                                                        ------

                                  [CloudFront-Policy, eyJTd...wIjo...
```

- Pour API plus de détails, consultez la section New- CFSignedCookie in AWS Tools for PowerShell Cmdlet Reference.

### `New-CFSignedUrl`

L'exemple de code suivant montre comment utiliser`New-CFSignedUrl`.

Outils pour PowerShell

Exemple 1 : crée une URL signée vers la ressource spécifiée à l'aide d'une politique prédéfinie. L'URL sera valide pendant une heure. Un objet System.Uri contenant l'URL signée est émis dans le pipeline.

```
$params = @{
 "ResourceUri"="https://cdn.example.com/index.html"
 "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
 "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
 "ExpiresOn"=(Get-Date).AddHours(1)
}
New-CFSignedUrl @params
```

Exemple 2 : crée une URL signée vers la ressource spécifiée à l'aide d'une politique personnalisée. L'URL sera valide dans 24 heures et expirera une semaine plus tard.

```
$start = (Get-Date).AddHours(24)
$params = @{
 "ResourceUri"="https://cdn.example.com/index.html"
 "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
 "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
 "ExpiresOn"=(Get-Date).AddDays(7)
    "ActiveFrom"=$start
}
New-CFSignedUrl @params
```

Exemple 3 : crée une URL signée vers la ressource spécifiée à l'aide d'une politique personnalisée. L'URL sera valide dans 24 heures et expirera une semaine plus tard. L'accès à la ressource est limité à la plage d'adresses IP spécifiée.

```
$start = (Get-Date).AddHours(24)
$params = @{
 "ResourceUri"="https://cdn.example.com/index.html"
 "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
 "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
 "ExpiresOn"=(Get-Date).AddDays(7)
```

```
    "ActiveFrom"=$start
    "IpRange"="192.0.2.0/24"
}
New-CFSignedUrl @params
```

- Pour API plus de détails, consultez la section New- CFSignedUrl in AWS Tools for PowerShell Cmdlet Reference.

# CloudTrail exemples utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS Tools for PowerShell with CloudTrail.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `Find-CTEvent`

L'exemple de code suivant montre comment utiliser`Find-CTEvent`.

Outils pour PowerShell

Exemple 1 : renvoie tous les événements survenus au cours des sept derniers jours. Par défaut, l'applet de commande effectue automatiquement plusieurs appels pour transmettre tous les événements, puis s'arrête lorsque le service indique qu'aucune autre donnée n'est disponible.

```
Find-CTEvent
```

Exemple 2 : renvoie tous les événements survenus au cours des sept derniers jours en spécifiant une région qui n'est pas la valeur par défaut du shell actuel.

```
Find-CTEvent -Region eu-central-1
```

Exemple 3 : renvoie tous les événements associés à l' RunInstances APIappel.

```
Find-CTEvent -LookupAttribute @{ AttributeKey="EventName";
 AttributeValue="RunInstances" }
```

Exemple 4 : renvoie les 5 premiers événements disponibles. Le jeton à utiliser pour récupérer d'autres événements est attaché au **`$AWSHistory.LastServiceResponse`** membre sous la forme d'une propriété de note nommée NextToken « ».

```
Find-CTEvent -MaxResult 5
```

Exemple 5 : renvoie les 10 événements suivants en utilisant le jeton « page suivante » d'un appel précédent pour indiquer par où commencer le renvoi des événements dans la séquence.

```
Find-CTEvent -MaxResult 10 -NextToken $AWSHistory.LastServiceResponse.NextToken
```

Exemple 6 : Cet exemple montre comment parcourir en boucle les événements disponibles à l'aide de la pagination manuelle, en récupérant un maximum de 5 événements par appel.

```
$nextToken = $null
do
{
    Find-CTEvent -MaxResult 5 -NextToken $nextToken
    $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Pour API plus de détails, consultez la section LookupEventsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CTTrail`

L'exemple de code suivant montre comment utiliser`Get-CTTrail`.

Outils pour PowerShell

Exemple 1 : renvoie les paramètres de tous les sentiers associés à la région actuelle pour votre compte.

```
Get-CTTrail
```

Exemple 2 : renvoie les paramètres pour les pistes spécifiées.

```
Get-CTTrail -TrailNameList trail1,trail2
```

Exemple 3 : renvoie les paramètres pour les sentiers spécifiés qui ont été créés dans une région autre que la région par défaut du shell actuel (dans ce cas, la région de Francfort (eu-central-1)).

```
Get-CTTrail -TrailNameList trailABC,trailDEF -Region eu-central-1
```

- Pour API plus de détails, consultez la section DescribeTrailsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CTTrailStatus`

L'exemple de code suivant montre comment utiliser`Get-CTTrailStatus`.

Outils pour PowerShell

Exemple 1 : renvoie les informations d'état du parcours portant le nom « myExampleTrail ». Les données renvoyées incluent des informations sur les erreurs de livraison, les erreurs Amazon SNS et Amazon S3, ainsi que les heures de début et de fin de journalisation du parcours. Cet exemple suppose que le sentier a été créé dans la même région que le shell par défaut actuel.

```
Get-CTTrailStatus -Name myExampleTrail
```

Exemple 2 : renvoie les informations d'état d'un parcours créé dans une région autre que la région par défaut du shell actuel (dans ce cas, la région de Francfort (eu-central-1)).

```
Get-CTTrailStatus -Name myExampleTrail -Region eu-central-1
```

- Pour API plus de détails, consultez la section GetTrailStatusRéférence des AWS Tools for PowerShell applets de commande.

### `New-CTTrail`

L'exemple de code suivant montre comment utiliser`New-CTTrail`.

Outils pour PowerShell

Exemple 1 : crée un journal qui utilisera le bucket « mycloudtrailbucket » pour le stockage des fichiers journaux.

```
New-CTTrail -Name "awscloudtrail-example" -S3BucketName "amzn-s3-demo-bucket"
```

Exemple 2 : crée un journal qui utilisera le bucket « mycloudtrailbucket » pour le stockage des fichiers journaux. Les objets S3 représentant les journaux auront un préfixe de clé commun « mylogs ». Lorsque de nouveaux journaux sont envoyés au bucket, une notification est envoyée à la SNS rubrique « mlog-deliverytopic ». Cet exemple utilise le splatting pour fournir les valeurs des paramètres à l'applet de commande.

```
$params = @{
    Name="awscloudtrail-example"
    S3BucketName="amzn-s3-demo-bucket"
    S3KeyPrefix="mylogs"
    SnsTopicName="mlog-deliverytopic"
}
New-CTTrail @params
```

- Pour API plus de détails, consultez la section CreateTrailRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-CTTrail`

L'exemple de code suivant montre comment utiliser`Remove-CTTrail`.

Outils pour PowerShell

Exemple 1 : Supprime le parcours spécifié. Vous serez invité à confirmer avant que la commande ne soit exécutée. Pour supprimer la confirmation, ajoutez le paramètre de commutation -Force.

```
Remove-CTTrail -Name "awscloudtrail-example"
```

- Pour API plus de détails, consultez la section DeleteTrailRéférence des AWS Tools for PowerShell applets de commande.

### `Start-CTLogging`

L'exemple de code suivant montre comment utiliser`Start-CTLogging`.

Outils pour PowerShell

Exemple 1 : Démarre l'enregistrement des AWS API appels et la livraison du fichier journal pour le journal nommé « myExampleTrail ». Cet exemple suppose que le sentier a été créé dans la même région que le shell par défaut actuel.

```
Start-CTLogging -Name myExampleTrail
```

Exemple 2 : Démarre l'enregistrement des AWS API appels et la livraison du fichier journal pour un journal créé dans une région autre que la région par défaut du shell actuel (dans ce cas, la région de Francfort (eu-central-1)).

```
Start-CTLogging -Name myExampleTrail -Region eu-central-1
```

- Pour API plus de détails, consultez la section StartLoggingRéférence des AWS Tools for PowerShell applets de commande.

### `Stop-CTLogging`

L'exemple de code suivant montre comment utiliser`Stop-CTLogging`.

Outils pour PowerShell

Exemple 1 : Suspend l'enregistrement des AWS API appels et la livraison du fichier journal pour le parcours nommé « myExampleTrail ». Cet exemple suppose que le sentier a été créé dans la même région que le shell par défaut actuel.

```
Stop-CTLogging -Name myExampleTrail
```

Exemple 2 : Suspend l'enregistrement des AWS API appels et la livraison du fichier journal pour un journal créé dans une région autre que la région par défaut du shell actuel (dans ce cas, la région de Francfort (eu-central-1)).

```
Stop-CTLogging -Name myExampleTrail -Region eu-central-1
```

- Pour API plus de détails, consultez la section StopLoggingRéférence des AWS Tools for PowerShell applets de commande.

### `Update-CTTrail`

L'exemple de code suivant montre comment utiliser`Update-CTTrail`.

Outils pour PowerShell

Exemple 1 : met à jour le journal spécifié afin que les événements de service globaux (tels que ceux provenant deIAM) soient enregistrés et remplace le préfixe de clé commun des fichiers journaux ultérieurs par « globallogs ».

```
Update-CTTrail -Name "awscloudtrail-example" -IncludeGlobalServiceEvents $true -
S3KeyPrefix "globallogs"
```

Exemple 2 : met à jour le journal spécifié afin que les notifications concernant les nouvelles livraisons de journaux soient envoyées au SNS sujet spécifié.

```
Update-CTTrail -Name "awscloudtrail-example" -SnsTopicName "mlog-deliverytopic2"
```

Exemple 3 : met à jour le journal spécifié afin que les journaux soient envoyés dans un autre compartiment.

```
Update-CTTrail -Name "awscloudtrail-example" -S3BucketName "otherlogs"
```

- Pour API plus de détails, consultez la section UpdateTrailRéférence des AWS Tools for PowerShell applets de commande.

## CloudWatch exemples utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS Tools for PowerShell with CloudWatch.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `Get-CWDashboard`

L'exemple de code suivant montre comment utiliser`Get-CWDashboard`.

Outils pour PowerShell

Exemple 1 : renvoie l'arn dans le corps du tableau de bord spécifié.

```
Get-CWDashboard -DashboardName Dashboard1
```

Sortie :

```
DashboardArn                                          DashboardBody
------------                                          -------------
arn:aws:cloudwatch::123456789012:dashboard/Dashboard1 {...
```

- Pour API plus de détails, consultez la section GetDashboardRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CWDashboardList`

L'exemple de code suivant montre comment utiliser`Get-CWDashboardList`.

Outils pour PowerShell

Exemple 1 : renvoie la collection de tableaux de bord pour votre compte.

```
Get-CWDashboardList
```

Sortie :

```
DashboardArn DashboardName LastModified        Size
```

```
------------ ------------ ------------        ----
arn:...      Dashboard1    7/6/2017 8:14:15 PM 252
```

Exemple 2 : renvoie la collection de tableaux de bord de votre compte dont le nom commence par le préfixe « dev ».

```
Get-CWDashboardList -DashboardNamePrefix dev
```

- Pour API plus de détails, consultez la section ListDashboardsRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-CWDashboard`

L'exemple de code suivant montre comment utiliser`Remove-CWDashboard`.

Outils pour PowerShell

Exemple 1 : Supprime le tableau de bord spécifié, en le promouvant pour confirmation avant de continuer. Pour contourner la confirmation, ajoutez le commutateur -Force à la commande.

```
Remove-CWDashboard -DashboardName Dashboard1
```

- Pour API plus de détails, consultez la section DeleteDashboardsRéférence des AWS Tools for PowerShell applets de commande.

### `Write-CWDashboard`

L'exemple de code suivant montre comment utiliser`Write-CWDashboard`.

Outils pour PowerShell

Exemple 1 : crée ou met à jour le tableau de bord nommé « Dashboard1 » pour inclure deux widgets métriques côte à côte.

```
$dashBody = @"
{
    "widgets":[
        {
             "type":"metric",
             "x":0,
             "y":0,
```

```
            "width":12,
            "height":6,
            "properties":{
               "metrics":[
                  [
                     "AWS/EC2",
                     "CPUUtilization",
                     "InstanceId",
                     "i-012345"
                  ]
               ],
               "period":300,
               "stat":"Average",
               "region":"us-east-1",
               "title":"EC2 Instance CPU"
            }
        },
        {
            "type":"metric",
            "x":12,
            "y":0,
            "width":12,
            "height":6,
            "properties":{
               "metrics":[
                  [
                     "AWS/S3",
                     "BucketSizeBytes",
                     "BucketName",
                     "amzn-s3-demo-bucket"
                  ]
               ],
               "period":86400,
               "stat":"Maximum",
               "region":"us-east-1",
               "title":"amzn-s3-demo-bucket bytes"
            }
        }
    ]
}
"@

Write-CWDashboard -DashboardName Dashboard1 -DashboardBody $dashBody
```

Exemple 2 : crée ou met à jour le tableau de bord, en redirigeant le contenu décrivant le tableau de bord vers l'applet de commande.

```
$dashBody = @"
{
...
}
"@

$dashBody | Write-CWDashboard -DashboardName Dashboard1
```

- Pour API plus de détails, consultez la section PutDashboardRéférence des AWS Tools for PowerShell applets de commande.

### `Write-CWMetricData`

L'exemple de code suivant montre comment utiliser`Write-CWMetricData`.

Outils pour PowerShell

Exemple 1 : crée un nouvel MetricDatum objet et l'écrit dans Amazon Web Services CloudWatch Metrics.

```
### Create a MetricDatum .NET object
$Metric = New-Object -TypeName Amazon.CloudWatch.Model.MetricDatum
$Metric.Timestamp = [DateTime]::UtcNow
$Metric.MetricName = 'CPU'
$Metric.Value = 50

### Write the metric data to the CloudWatch service
Write-CWMetricData -Namespace instance1 -MetricData $Metric
```

- Pour API plus de détails, consultez la section PutMetricDataRéférence des AWS Tools for PowerShell applets de commande.

## CodeCommit exemples utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS Tools for PowerShell with CodeCommit.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques



## Actions

### `Get-CCBranch`

L'exemple de code suivant montre comment utiliser`Get-CCBranch`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir des informations sur la branche spécifiée pour le référentiel spécifié.

```
Get-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch
```

Sortie :

```
BranchName                              CommitId
----------                              --------
MyNewBranch                             7763222d...561fc9c9
```

- Pour API plus de détails, consultez la section GetBranchRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CCBranchList`

L'exemple de code suivant montre comment utiliser`Get-CCBranchList`.

Outils pour PowerShell

Exemple 1 : Cet exemple obtient une liste des noms de branches pour le référentiel spécifié.

```
Get-CCBranchList -RepositoryName MyDemoRepo
```

Sortie :

```
master
MyNewBranch
```

- Pour API plus de détails, consultez la section ListBranchesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CCRepository`

L'exemple de code suivant montre comment utiliser`Get-CCRepository`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir des informations pour le référentiel spécifié.

```
Get-CCRepository -RepositoryName MyDemoRepo
```

Sortie :

```
AccountId             : 80398EXAMPLE
Arn                   : arn:aws:codecommit:us-east-1:80398EXAMPLE:MyDemoRepo
CloneUrlHttp          : https://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CloneUrlSsh           : ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CreationDate          : 9/8/2015 3:21:33 PM
DefaultBranch         :
LastModifiedDate      : 9/8/2015 3:21:33 PM
RepositoryDescription : This is a repository for demonstration purposes.
RepositoryId          : c7d0d2b0-ce40-4303-b4c3-38529EXAMPLE
RepositoryName        : MyDemoRepo
```

- Pour API plus de détails, consultez la section GetRepositoryRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CCRepositoryBatch`

L'exemple de code suivant montre comment utiliser`Get-CCRepositoryBatch`.

Outils pour PowerShell

Exemple 1 : Cet exemple confirme lesquels des référentiels spécifiés sont trouvés et non trouvés.

```
Get-CCRepositoryBatch -RepositoryName MyDemoRepo, MyNewRepo, AMissingRepo
```

Sortie :

```
Repositories                            RepositoriesNotFound
------------                            --------------------
{MyDemoRepo, MyNewRepo}                {AMissingRepo}
```

- Pour API plus de détails, consultez la section BatchGetRepositoriesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CCRepositoryList`

L'exemple de code suivant montre comment utiliser`Get-CCRepositoryList`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie tous les référentiels par ordre croissant par nom de référentiel.

```
Get-CCRepositoryList -Order Ascending -SortBy RepositoryName
```

Sortie :

```
RepositoryId                            RepositoryName
------------                            --------------
c7d0d2b0-ce40-4303-b4c3-38529EXAMPLE    MyDemoRepo
05f30c66-e3e3-4f91-a0cd-1c84aEXAMPLE    MyNewRepo
```

- Pour API plus de détails, consultez la section ListRepositoriesRéférence des AWS Tools for PowerShell applets de commande.

### `New-CCBranch`

L'exemple de code suivant montre comment utiliser`New-CCBranch`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une nouvelle branche avec le nom spécifié pour le référentiel spécifié et l'ID de validation spécifié.

```
New-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch -CommitId
 7763222d...561fc9c9
```

- Pour API plus de détails, consultez la section CreateBranchRéférence des AWS Tools for PowerShell applets de commande.

### `New-CCRepository`

L'exemple de code suivant montre comment utiliser`New-CCRepository`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un nouveau référentiel avec le nom et la description spécifiés.

```
New-CCRepository -RepositoryName MyDemoRepo -RepositoryDescription "This is a
 repository for demonstration purposes."
```

Sortie :

```
AccountId             : 80398EXAMPLE
Arn                   : arn:aws:codecommit:us-east-1:80398EXAMPLE:MyDemoRepo
CloneUrlHttp          : https://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CloneUrlSsh           : ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CreationDate          : 9/18/2015 4:13:25 PM
DefaultBranch         :
LastModifiedDate      : 9/18/2015 4:13:25 PM
RepositoryDescription : This is a repository for demonstration purposes.
RepositoryId          : 43ef2443-3372-4b12-9e78-65c27EXAMPLE
RepositoryName        : MyDemoRepo
```

- Pour API plus de détails, consultez la section CreateRepositoryRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-CCRepository`

L'exemple de code suivant montre comment utiliser`Remove-CCRepository`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime de force le référentiel spécifié. La commande vous demandera une confirmation avant de continuer. Ajoutez le paramètre -Force pour supprimer le référentiel sans invite.

```
Remove-CCRepository -RepositoryName MyDemoRepo
```

Sortie :

```
43ef2443-3372-4b12-9e78-65c27EXAMPLE
```

- Pour API plus de détails, consultez la section DeleteRepositoryRéférence des AWS Tools for PowerShell applets de commande.

### `Update-CCDefaultBranch`

L'exemple de code suivant montre comment utiliser`Update-CCDefaultBranch`.

Outils pour PowerShell

Exemple 1 : Cet exemple remplace la branche par défaut du référentiel spécifié par la branche spécifiée.

```
Update-CCDefaultBranch -RepositoryName MyDemoRepo -DefaultBranchName MyNewBranch
```

- Pour API plus de détails, consultez la section UpdateDefaultBranchRéférence des AWS Tools for PowerShell applets de commande.

### `Update-CCRepositoryDescription`

L'exemple de code suivant montre comment utiliser`Update-CCRepositoryDescription`.

Outils pour PowerShell

Exemple 1 : Cet exemple modifie la description du référentiel spécifié.

```
Update-CCRepositoryDescription -RepositoryName MyDemoRepo -RepositoryDescription
 "This is an updated description."
```

- Pour API plus de détails, consultez la section UpdateRepositoryDescriptionRéférence des AWS Tools for PowerShell applets de commande.

**`Update-CCRepositoryName`**

L'exemple de code suivant montre comment utiliser`Update-CCRepositoryName`.

Outils pour PowerShell

Exemple 1 : Cet exemple modifie le nom du référentiel spécifié.

```
Update-CCRepositoryName -NewName MyDemoRepo2 -OldName MyDemoRepo
```

- Pour API plus de détails, consultez la section UpdateRepositoryNameRéférence des AWS Tools for PowerShell applets de commande.

# CodeDeploy exemples utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS Tools for PowerShell with CodeDeploy.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

# Actions

### `Add-CDOnPremiseInstanceTag`

L'exemple de code suivant montre comment utiliser`Add-CDOnPremiseInstanceTag`.

Outils pour PowerShell

Exemple 1 : Cet exemple ajoute une balise d'instance locale avec la clé et la valeur spécifiées pour l'instance locale spécifiée.

```
Add-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" = "Name";
 "Value" = "CodeDeployDemo-OnPrem"}
```

- Pour API plus de détails, consultez la section AddTagsToOnPremisesInstancesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CDApplication`

L'exemple de code suivant montre comment utiliser`Get-CDApplication`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir des informations sur l'application spécifiée.

```
Get-CDApplication -ApplicationName CodeDeployDemoApplication
```

Sortie :

```
ApplicationId                           ApplicationName             CreateTime
         LinkedToGitHub
-------------                           ---------------             ----------
         --------------
e07fb938-091e-4f2f-8963-4d3e8EXAMPLE    CodeDeployDemoApplication   7/20/2015
 9:49:48 PM    False
```

- Pour API plus de détails, consultez la section GetApplicationRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CDApplicationBatch`

L'exemple de code suivant montre comment utiliser`Get-CDApplicationBatch`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir des informations sur les applications spécifiées.

```
Get-CDApplicationBatch -ApplicationName CodeDeployDemoApplication,
 CodePipelineDemoApplication
```

Sortie :

```
ApplicationId                           ApplicationName              CreateTime
         LinkedToGitHub
-------------                           ---------------              ----------
         --------------
e07fb938-091e-4f2f-8963-4d3e8EXAMPLE    CodeDeployDemoApplication    7/20/2015
 9:49:48 PM    False
1ecfd602-62f1-4038-8f0d-06688EXAMPLE    CodePipelineDemoApplication  8/13/2015
 5:53:26 PM    False
```

- Pour API plus de détails, consultez la section BatchGetApplicationsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CDApplicationList`

L'exemple de code suivant montre comment utiliser`Get-CDApplicationList`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir une liste des applications disponibles.

```
Get-CDApplicationList
```

Sortie :

```
CodeDeployDemoApplication
CodePipelineDemoApplication
```

- Pour API plus de détails, consultez la section ListApplicationsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CDApplicationRevision`

L'exemple de code suivant montre comment utiliser`Get-CDApplicationRevision`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir des informations sur la révision de l'application spécifiée.

```
$revision = Get-CDApplicationRevision -ApplicationName CodeDeployDemoApplication
 -S3Location_Bucket amzn-s3-demo-bucket -Revision_RevisionType S3 -
S3Location_Key 5xd27EX.zip -S3Location_BundleType zip -S3Location_ETag
 4565c1ac97187f190c1a90265EXAMPLE
Write-Output ("Description = " + $revision.RevisionInfo.Description + ",
 RegisterTime = " + $revision.RevisionInfo.RegisterTime)
```

Sortie :

```
Description = Application revision registered by Deployment ID: d-CX9CHN3EX,
 RegisterTime = 07/20/2015 23:46:42
```

- Pour API plus de détails, consultez la section GetApplicationRevisionRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CDApplicationRevisionList`

L'exemple de code suivant montre comment utiliser`Get-CDApplicationRevisionList`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir des informations sur les révisions disponibles pour l'application spécifiée.

```
ForEach ($revision in (Get-CDApplicationRevisionList -ApplicationName
 CodeDeployDemoApplication -Deployed Ignore)) {
>>   If ($revision.RevisionType -Eq "S3") {
```

```
>>     Write-Output ("Type = S3, Bucket = " + $revision.S3Location.Bucket
 + ", BundleType = " + $revision.S3Location.BundleType + ", ETag = " +
 $revision.S3Location.ETag + ", Key = " + $revision.S3Location.Key)
>>   }
>>   If ($revision.RevisionType -Eq "GitHub") {
>>     Write-Output ("Type = GitHub, CommitId = " +
 $revision.GitHubLocation.CommitId + ", Repository = " +
 $revision.GitHubLocation.Repository)
>>   }
>> }
>>
```

Sortie :

```
Type = S3, Bucket = MyBucket, BundleType = zip, ETag =
 4565c1ac97187f190c1a90265EXAMPLE, Key = 5xd27EX.zip
Type = GitHub, CommitId = f48933c3...76405362, Repository = MyGitHubUser/
CodeDeployDemoRepo
```

- Pour API plus de détails, consultez la section ListApplicationRevisionsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CDDeployment`

L'exemple de code suivant montre comment utiliser`Get-CDDeployment`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir des informations récapitulatives sur le déploiement spécifié.

```
Get-CDDeployment -DeploymentId d-QZMRGSTEX
```

Sortie :

```
ApplicationName              : CodeDeployDemoApplication
CompleteTime                 : 7/23/2015 11:26:04 PM
CreateTime                   : 7/23/2015 11:24:43 PM
Creator                      : user
DeploymentConfigName         : CodeDeployDefault.OneAtATime
DeploymentGroupName          : CodeDeployDemoFleet
DeploymentId                 : d-QZMRGSTEX
```

```
DeploymentOverview            : Amazon.CodeDeploy.Model.DeploymentOverview
Description                   :
ErrorInformation              :
IgnoreApplicationStopFailures : False
Revision                      : Amazon.CodeDeploy.Model.RevisionLocation
StartTime                     : 1/1/0001 12:00:00 AM
Status                        : Succeeded
```

Exemple 2 : Cet exemple permet d'obtenir des informations sur le statut des instances participant au déploiement spécifié.

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).DeploymentOverview
```

Sortie :

```
Failed     : 0
InProgress : 0
Pending    : 0
Skipped    : 0
Succeeded  : 3
```

Exemple 3 : Cet exemple permet d'obtenir des informations sur la révision de l'application pour le déploiement spécifié.

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).Revision.S3Location
```

Sortie :

```
Bucket     : MyBucket
BundleType : zip
ETag       : cfbb81b304ee5e27efc21adaed3EXAMPLE
Key        : clzfqEX
Version    :
```

- Pour API plus de détails, consultez la section GetDeploymentRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CDDeploymentBatch`

L'exemple de code suivant montre comment utiliser`Get-CDDeploymentBatch`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir des informations sur les déploiements spécifiés.

```
Get-CDDeploymentBatch -DeploymentId d-QZMRGSTEX, d-RR0T5KTEX
```

Sortie :

```
ApplicationName               : CodeDeployDemoApplication
CompleteTime                  : 7/23/2015 11:26:04 PM
CreateTime                    : 7/23/2015 11:24:43 PM
Creator                       : user
DeploymentConfigName          : CodeDeployDefault.OneAtATime
DeploymentGroupName           : CodeDeployDemoFleet
DeploymentId                  : d-QZMRGSTEX
DeploymentOverview            : Amazon.CodeDeploy.Model.DeploymentOverview
Description                   :
ErrorInformation              :
IgnoreApplicationStopFailures : False
Revision                      : Amazon.CodeDeploy.Model.RevisionLocation
StartTime                     : 1/1/0001 12:00:00 AM
Status                        : Succeeded

ApplicationName               : CodePipelineDemoApplication
CompleteTime                  : 7/23/2015 6:07:30 PM
CreateTime                    : 7/23/2015 6:06:29 PM
Creator                       : user
DeploymentConfigName          : CodeDeployDefault.OneAtATime
DeploymentGroupName           : CodePipelineDemoFleet
DeploymentId                  : d-RR0T5KTEX
DeploymentOverview            : Amazon.CodeDeploy.Model.DeploymentOverview
Description                   :
ErrorInformation              :
IgnoreApplicationStopFailures : False
Revision                      : Amazon.CodeDeploy.Model.RevisionLocation
StartTime                     : 1/1/0001 12:00:00 AM
Status                        : Succeeded
```

- Pour API plus de détails, consultez la section BatchGetDeploymentsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CDDeploymentConfig`

L'exemple de code suivant montre comment utiliser`Get-CDDeploymentConfig`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir des informations récapitulatives sur la configuration de déploiement spécifiée.

```
Get-CDDeploymentConfig -DeploymentConfigName ThreeQuartersHealthy
```

Sortie :

```
CreateTime              DeploymentConfigId                      DeploymentConfigName
    MinimumHealthyHosts
----------              ------------------                      --------------------
    -------------------
10/3/2014 4:32:30 PM    518a3950-d034-46a1-9d2c-3c949EXAMPLE    ThreeQuartersHealthy
    Amazon.CodeDeploy.Model.MinimumHealthyHosts
```

Exemple 2 : Cet exemple permet d'obtenir des informations sur la définition de la configuration de déploiement spécifiée.

```
Write-Output ((Get-CDDeploymentConfig -DeploymentConfigName
 ThreeQuartersHealthy).MinimumHealthyHosts)
```

Sortie :

```
Type            Value
----            -----
FLEET_PERCENT   75
```

- Pour API plus de détails, consultez la section GetDeploymentConfigRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CDDeploymentConfigList`

L'exemple de code suivant montre comment utiliser`Get-CDDeploymentConfigList`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir une liste des configurations de déploiement disponibles.

```
Get-CDDeploymentConfigList
```

Sortie :

```
ThreeQuartersHealthy
CodeDeployDefault.OneAtATime
CodeDeployDefault.AllAtOnce
CodeDeployDefault.HalfAtATime
```

- Pour API plus de détails, consultez la section ListDeploymentConfigsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CDDeploymentGroup`

L'exemple de code suivant montre comment utiliser`Get-CDDeploymentGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir des informations sur le groupe de déploiement spécifié.

```
Get-CDDeploymentGroup -ApplicationName CodeDeployDemoApplication -
DeploymentGroupName CodeDeployDemoFleet
```

Sortie :

```
ApplicationName              : CodeDeployDemoApplication
AutoScalingGroups            : {}
DeploymentConfigName         : CodeDeployDefault.OneAtATime
DeploymentGroupId            : 7d7c098a-b444-4b27-96ef-22791EXAMPLE
DeploymentGroupName          : CodeDeployDemoFleet
Ec2TagFilters                : {Name}
OnPremisesInstanceTagFilters : {}
ServiceRoleArn               : arn:aws:iam::80398EXAMPLE:role/
CodeDeploySampleStack-4ph6EX-CodeDeployTrustRole-O9MWP7XTL8EX
TargetRevision               : Amazon.CodeDeploy.Model.RevisionLocation
```

- Pour API plus de détails, consultez la section GetDeploymentGroupRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CDDeploymentGroupList`

L'exemple de code suivant montre comment utiliser`Get-CDDeploymentGroupList`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir une liste de groupes de déploiement pour l'application spécifiée.

```
Get-CDDeploymentGroupList -ApplicationName CodeDeployDemoApplication
```

Sortie :

```
ApplicationName              DeploymentGroups
 NextToken
---------------              ----------------
 ---------
CodeDeployDemoApplication    {CodeDeployDemoFleet, CodeDeployProductionFleet}
```

- Pour API plus de détails, consultez la section ListDeploymentGroupsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CDDeploymentInstance`

L'exemple de code suivant montre comment utiliser`Get-CDDeploymentInstance`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir des informations sur l'instance spécifiée pour le déploiement spécifié.

```
Get-CDDeploymentInstance -DeploymentId d-QZMRGSTEX -InstanceId i-254e22EX
```

Sortie :

```
DeploymentId    : d-QZMRGSTEX
```

```
InstanceId      : arn:aws:ec2:us-east-1:80398EXAMPLE:instance/i-254e22EX
LastUpdatedAt   : 7/23/2015 11:25:24 PM
LifecycleEvents : {ApplicationStop, DownloadBundle, BeforeInstall, Install...}
Status          : Succeeded
```

- Pour API plus de détails, consultez la section GetDeploymentInstanceRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CDDeploymentInstanceList`

L'exemple de code suivant montre comment utiliser`Get-CDDeploymentInstanceList`.

Outils pour PowerShell

Exemple 1 : Cet exemple obtient une liste d'instances IDs pour le déploiement spécifié.

```
Get-CDDeploymentInstanceList -DeploymentId d-QZMRGSTEX
```

Sortie :

```
i-254e22EX
i-274e22EX
i-3b4e22EX
```

- Pour API plus de détails, consultez la section ListDeploymentInstancesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CDDeploymentList`

L'exemple de code suivant montre comment utiliser`Get-CDDeploymentList`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir une liste des déploiements IDs pour l'application et le groupe de déploiement spécifiés.

```
Get-CDDeploymentList -ApplicationName CodeDeployDemoApplication -DeploymentGroupName
 CodeDeployDemoFleet
```

Sortie :

```
d-QZMRGSTEX
d-RR0T5KTEX
```

- Pour API plus de détails, consultez la section ListDeploymentsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CDOnPremiseInstance`

L'exemple de code suivant montre comment utiliser`Get-CDOnPremiseInstance`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir des informations sur l'instance locale spécifiée.

```
Get-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

Sortie :

```
DeregisterTime : 1/1/0001 12:00:00 AM
IamUserArn     : arn:aws:iam::80398EXAMPLE:user/CodeDeployDemoUser
InstanceArn    : arn:aws:codedeploy:us-east-1:80398EXAMPLE:instance/
AssetTag12010298EX_rDH556dxEX
InstanceName   : AssetTag12010298EX
RegisterTime   : 4/3/2015 6:36:24 PM
Tags           : {Name}
```

- Pour API plus de détails, consultez la section GetOnPremisesInstanceRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CDOnPremiseInstanceBatch`

L'exemple de code suivant montre comment utiliser`Get-CDOnPremiseInstanceBatch`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir des informations sur les instances locales spécifiées.

```
Get-CDOnPremiseInstanceBatch -InstanceName AssetTag12010298EX, AssetTag12010298EX-2
```

Sortie :

```
DeregisterTime : 1/1/0001 12:00:00 AM
IamUserArn     : arn:aws:iam::80398EXAMPLE:user/CodeDeployFRWUser
InstanceArn    : arn:aws:codedeploy:us-east-1:80398EXAMPLE:instance/
AssetTag12010298EX-2_XmeSz18rEX
InstanceName   : AssetTag12010298EX-2
RegisterTime   : 4/3/2015 6:38:52 PM
Tags           : {Name}

DeregisterTime : 1/1/0001 12:00:00 AM
IamUserArn     : arn:aws:iam::80398EXAMPLE:user/CodeDeployDemoUser
InstanceArn    : arn:aws:codedeploy:us-east-1:80398EXAMPLE:instance/
AssetTag12010298EX_rDH556dxEX
InstanceName   : AssetTag12010298EX
RegisterTime   : 4/3/2015 6:36:24 PM
Tags           : {Name}
```

- Pour API plus de détails, consultez la section BatchGetOnPremisesInstancesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CDOnPremiseInstanceList`

L'exemple de code suivant montre comment utiliser`Get-CDOnPremiseInstanceList`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir une liste des noms d'instances locales disponibles.

```
Get-CDOnPremiseInstanceList
```

Sortie :

```
AssetTag12010298EX
AssetTag12010298EX-2
```

- Pour API plus de détails, consultez la section ListOnPremisesInstancesRéférence des AWS Tools for PowerShell applets de commande.

### `New-CDApplication`

L'exemple de code suivant montre comment utiliser`New-CDApplication`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une nouvelle application portant le nom spécifié.

```
New-CDApplication -ApplicationName MyNewApplication
```

Sortie :

```
f19e4b61-2231-4328-b0fd-e57f5EXAMPLE
```

- Pour API plus de détails, consultez la section CreateApplicationRéférence des AWS Tools for PowerShell applets de commande.

### `New-CDDeployment`

L'exemple de code suivant montre comment utiliser`New-CDDeployment`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un nouveau déploiement pour l'application et le groupe de déploiement spécifiés avec la configuration de déploiement et la révision de l'application spécifiées.

```
New-CDDeployment -ApplicationName MyNewApplication -S3Location_Bucket amzn-s3-demo-
bucket -S3Location_BundleType zip -DeploymentConfigName CodeDeployDefault.OneAtATime
 -DeploymentGroupName MyNewDeploymentGroup -IgnoreApplicationStopFailures $True -
S3Location_Key aws-codedeploy_linux-master.zip -RevisionType S3
```

Sortie :

```
d-ZHROG7UEX
```

Exemple 2 : Cet exemple montre comment spécifier des groupes de balises d'EC2instance par lesquels une instance doit être identifiée afin qu'elle soit incluse dans l'environnement de remplacement pour un déploiement bleu/vert.

```
New-CDDeployment -ApplicationName MyNewApplication -S3Location_Bucket amzn-s3-demo-
bucket -S3Location_BundleType zip -DeploymentConfigName CodeDeployDefault.OneAtATime
 -DeploymentGroupName MyNewDeploymentGroup -IgnoreApplicationStopFailures $True
```

```
 -S3Location_Key aws-codedeploy_linux-master.zip -RevisionType S3 -Ec2TagSetList
 @(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{Key=
```

Sortie :

```
d-ZHROG7UEX
```

- Pour API plus de détails, consultez la section CreateDeploymentRéférence des AWS Tools for PowerShell applets de commande.

### `New-CDDeploymentConfig`

L'exemple de code suivant montre comment utiliser`New-CDDeploymentConfig`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une nouvelle configuration de déploiement avec le nom et le comportement spécifiés.

```
New-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts -
MinimumHealthyHosts_Type HOST_COUNT -MinimumHealthyHosts_Value 2
```

Sortie :

```
0f3e8187-44ef-42da-aeed-b6823EXAMPLE
```

- Pour API plus de détails, consultez la section CreateDeploymentConfigRéférence des AWS Tools for PowerShell applets de commande.

### `New-CDDeploymentGroup`

L'exemple de code suivant montre comment utiliser`New-CDDeploymentGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un groupe de déploiement avec le nom spécifié, le groupe Auto Scaling, la configuration de déploiement, le tag et le rôle de service, pour l'application spécifiée.

```
New-CDDeploymentGroup -ApplicationName MyNewApplication -AutoScalingGroup
 CodeDeployDemo-ASG -DeploymentConfigName CodeDeployDefault.OneAtATime
```

```
 -DeploymentGroupName MyNewDeploymentGroup -Ec2TagFilter @{Key="Name";
 Type="KEY_AND_VALUE"; Value="CodeDeployDemo"} -ServiceRoleArn
 arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo
```

Sortie :

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

Exemple 2 : Cet exemple montre comment spécifier des groupes de balises d'EC2instance par lesquels une instance doit être identifiée afin qu'elle soit incluse dans l'environnement de remplacement pour un déploiement bleu/vert.

```
New-CDDeploymentGroup -ApplicationName MyNewApplication -AutoScalingGroup
 CodeDeployDemo-ASG -DeploymentConfigName CodeDeployDefault.OneAtATime
 -DeploymentGroupName MyNewDeploymentGroup -Ec2TagFilter @{Key="Name";
 Type="KEY_AND_VALUE"; Value="CodeDeployDemo"} -ServiceRoleArn
 arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo -Ec2TagSetList
 @(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{Key=
```

Sortie :

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

- Pour API plus de détails, consultez la section CreateDeploymentGroupRéférence des AWS Tools for PowerShell applets de commande.

### `Register-CDApplicationRevision`

L'exemple de code suivant montre comment utiliser`Register-CDApplicationRevision`.

Outils pour PowerShell

Exemple 1 : Cet exemple enregistre une révision d'application à l'emplacement Amazon S3 spécifié, pour l'application spécifiée.

```
Register-CDApplicationRevision -ApplicationName MyNewApplication -S3Location_Bucket
 amzn-s3-demo-bucket -S3Location_BundleType zip -S3Location_Key aws-
codedeploy_linux-master.zip -Revision_RevisionType S3
```

- Pour API plus de détails, consultez la section RegisterApplicationRevisionRéférence des AWS Tools for PowerShell applets de commande.

### `Register-CDOnPremiseInstance`

L'exemple de code suivant montre comment utiliser`Register-CDOnPremiseInstance`.

Outils pour PowerShell

Exemple 1 : Cet exemple enregistre une instance locale avec le nom et l'IAMutilisateur spécifiés.

```
Register-CDOnPremiseInstance -IamUserArn arn:aws:iam::80398EXAMPLE:user/
CodeDeployDemoUser -InstanceName AssetTag12010298EX
```

- Pour API plus de détails, consultez la section RegisterOnPremisesInstanceRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-CDApplication`

L'exemple de code suivant montre comment utiliser`Remove-CDApplication`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime l'application portant le nom spécifié. La commande vous demandera une confirmation avant de continuer. Ajoutez le paramètre -Force pour supprimer l'application sans y être invité.

```
Remove-CDApplication -ApplicationName MyNewApplication
```

- Pour API plus de détails, consultez la section DeleteApplicationRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-CDDeploymentConfig`

L'exemple de code suivant montre comment utiliser`Remove-CDDeploymentConfig`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la configuration de déploiement portant le nom spécifié. La commande vous demandera une confirmation avant de continuer. Ajoutez le paramètre -Force pour supprimer la configuration de déploiement sans invite.

```
Remove-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts
```

- Pour API plus de détails, consultez la section DeleteDeploymentConfigRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-CDDeploymentGroup`

L'exemple de code suivant montre comment utiliser`Remove-CDDeploymentGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le groupe de déploiement portant le nom spécifié pour l'application spécifiée. La commande vous demandera une confirmation avant de continuer. Ajoutez le paramètre -Force pour supprimer le groupe de déploiement sans invite.

```
Remove-CDDeploymentGroup -ApplicationName MyNewApplication -DeploymentGroupName
 MyNewDeploymentGroup
```

- Pour API plus de détails, consultez la section DeleteDeploymentGroupRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-CDOnPremiseInstanceTag`

L'exemple de code suivant montre comment utiliser`Remove-CDOnPremiseInstanceTag`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la balise spécifiée pour l'instance locale portant le nom spécifié. La commande vous demandera une confirmation avant de continuer. Ajoutez le paramètre -Force pour supprimer le tag sans demande.

```
Remove-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" =
 "Name"; "Value" = "CodeDeployDemo-OnPrem"}
```

- Pour API plus de détails, consultez la section RemoveTagsFromOnPremisesInstancesRéférence des AWS Tools for PowerShell applets de commande.

### `Stop-CDDeployment`

L'exemple de code suivant montre comment utiliser`Stop-CDDeployment`.

Outils pour PowerShell

Exemple 1 : Cet exemple tente d'arrêter le déploiement avec l'ID de déploiement spécifié.

```
Stop-CDDeployment -DeploymentId d-LJQNREYEX
```

Sortie :

```
Status     StatusMessage
------     -------------
Pending    Stopping Pending. Stopping to schedule commands in the deployment
 instances
```

- Pour API plus de détails, consultez la section StopDeploymentRéférence des AWS Tools for PowerShell applets de commande.

### `Unregister-CDOnPremiseInstance`

L'exemple de code suivant montre comment utiliser`Unregister-CDOnPremiseInstance`.

Outils pour PowerShell

Exemple 1 : Cet exemple annule l'enregistrement de l'instance locale avec le nom spécifié.

```
Unregister-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

- Pour API plus de détails, consultez la section DeregisterOnPremisesInstanceRéférence des AWS Tools for PowerShell applets de commande.

### `Update-CDApplication`

L'exemple de code suivant montre comment utiliser`Update-CDApplication`.

Outils pour PowerShell

Exemple 1 : Cet exemple modifie le nom de l'application spécifiée.

```
Update-CDApplication -ApplicationName MyNewApplication -NewApplicationName
 MyNewApplication-2
```

- Pour API plus de détails, consultez la section UpdateApplicationRéférence des AWS Tools for PowerShell applets de commande.

### `Update-CDDeploymentGroup`

L'exemple de code suivant montre comment utiliser`Update-CDDeploymentGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple modifie le nom du groupe de déploiement spécifié pour l'application spécifiée.

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication -
CurrentDeploymentGroupName MyNewDeploymentGroup -NewDeploymentGroupName
 MyNewDeploymentGroup-2
```

Exemple 2 : Cet exemple montre comment spécifier des groupes de balises d'EC2instance par lesquels une instance doit être identifiée afin qu'elle soit incluse dans l'environnement de remplacement pour un déploiement bleu/vert.

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication -
CurrentDeploymentGroupName MyNewDeploymentGroup -NewDeploymentGroupName
 MyNewDeploymentGroup-2 -Ec2TagSetList
 @(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{Key=
```

- Pour API plus de détails, consultez la section UpdateDeploymentGroupRéférence des AWS Tools for PowerShell applets de commande.

## CodePipeline exemples utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS Tools for PowerShell with CodePipeline.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques



## Actions

### `Confirm-CPJob`

L'exemple de code suivant montre comment utiliser`Confirm-CPJob`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir le statut de la tâche spécifiée.

```
Confirm-CPJob -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE -Nonce 3
```

Sortie :

```
Value
-----
InProgress
```

- Pour API plus de détails, consultez la section AcknowledgeJobRéférence des AWS Tools for PowerShell applets de commande.

### `Disable-CPStageTransition`

L'exemple de code suivant montre comment utiliser`Disable-CPStageTransition`.

Outils pour PowerShell

Exemple 1 : Cet exemple désactive la transition entrante pour l'étape spécifiée dans le pipeline spécifié.

```
Disable-CPStageTransition -PipelineName CodePipelineDemo -Reason "Disabling
 temporarily." -StageName Beta -TransitionType Inbound
```

- Pour API plus de détails, consultez la section DisableStageTransitionRéférence des AWS Tools for PowerShell applets de commande.

### `Enable-CPStageTransition`

L'exemple de code suivant montre comment utiliser`Enable-CPStageTransition`.

Outils pour PowerShell

Exemple 1 : Cet exemple active la transition entrante pour l'étape spécifiée dans le pipeline spécifié.

```
Enable-CPStageTransition -PipelineName CodePipelineDemo -StageName Beta -
TransitionType Inbound
```

- Pour API plus de détails, consultez la section EnableStageTransitionRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CPActionType`

L'exemple de code suivant montre comment utiliser`Get-CPActionType`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir des informations sur toutes les actions disponibles pour le propriétaire spécifié.

```
ForEach ($actionType in (Get-CPActionType -ActionOwnerFilter AWS)) {
  Write-Output ("For Category = " + $actionType.Id.Category + ", Owner = " +
 $actionType.Id.Owner + ", Provider = " + $actionType.Id.Provider + ", Version = " +
 $actionType.Id.Version + ":")
  Write-Output ("  ActionConfigurationProperties:")
  ForEach ($acp in $actionType.ActionConfigurationProperties) {
    Write-Output ("    For " + $acp.Name + ":")
    Write-Output ("      Description = " + $acp.Description)
    Write-Output ("      Key = " + $acp.Key)
    Write-Output ("      Queryable = " + $acp.Queryable)
    Write-Output ("      Required = " + $acp.Required)
```

```
    Write-Output ("      Secret = " + $acp.Secret)
  }
  Write-Output ("  InputArtifactDetails:")
  Write-Output ("    MaximumCount = " +
 $actionType.InputArtifactDetails.MaximumCount)
  Write-Output ("    MinimumCount = " +
 $actionType.InputArtifactDetails.MinimumCount)
  Write-Output ("  OutputArtifactDetails:")
  Write-Output ("    MaximumCount = " +
 $actionType.OutputArtifactDetails.MaximumCount)
  Write-Output ("    MinimumCount = " +
 $actionType.OutputArtifactDetails.MinimumCount)
  Write-Output ("  Settings:")
  Write-Output ("    EntityUrlTemplate = " + $actionType.Settings.EntityUrlTemplate)
  Write-Output ("    ExecutionUrlTemplate = " +
 $actionType.Settings.ExecutionUrlTemplate)
}
```

Sortie :

```
For Category = Deploy, Owner = AWS, Provider = ElasticBeanstalk, Version = 1:
  ActionConfigurationProperties:
    For ApplicationName:
      Description = The AWS Elastic Beanstalk Application name
      Key = True
      Queryable = False
      Required = True
      Secret = False
    For EnvironmentName:
      Description = The AWS Elastic Beanstalk Environment name
      Key = True
      Queryable = False
      Required = True
      Secret = False
  InputArtifactDetails:
    MaximumCount = 1
    MinimumCount = 1
  OutputArtifactDetails:
    MaximumCount = 0
    MinimumCount = 0
  Settings:
    EntityUrlTemplate = https://console.aws.amazon.com/elasticbeanstalk/r/
application/{Config:ApplicationName}
```

```
    ExecutionUrlTemplate = https://console.aws.amazon.com/elasticbeanstalk/r/
application/{Config:ApplicationName}
For Category = Deploy, Owner = AWS, Provider = CodeDeploy, Version = 1:
  ActionConfigurationProperties:
    For ApplicationName:
      Description = The AWS CodeDeploy Application name
      Key = True
      Queryable = False
      Required = True
      Secret = False
    For DeploymentGroupName:
      Description = The AWS CodeDeploy Deployment Group name
      Key = True
      Queryable = False
      Required = True
      Secret = False
  InputArtifactDetails:
    MaximumCount = 1
    MinimumCount = 1
  OutputArtifactDetails:
    MaximumCount = 0
    MinimumCount = 0
  Settings:
    EntityUrlTemplate = https://console.aws.amazon.com/codedeploy/home?#/
applications/{Config:ApplicationName}/deployment-groups/{Config:DeploymentGroupName}
    ExecutionUrlTemplate = https://console.aws.amazon.com/codedeploy/home?#/
deployments/{ExternalExecutionId}
```

- Pour API plus de détails, consultez la section ListActionTypesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CPActionableJobList`

L'exemple de code suivant montre comment utiliser`Get-CPActionableJobList`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir des informations sur toutes les tâches exploitables pour la catégorie d'action, le propriétaire, le fournisseur, la version et les paramètres de requête spécifiés.

```
Get-CPActionableJobList -ActionTypeId_Category Build -ActionTypeId_Owner Custom
 -ActionTypeId_Provider MyCustomProviderName -ActionTypeId_Version 1 -QueryParam
 @{"ProjectName" = "MyProjectName"}
```

Sortie :

```
AccountId       Data                                Id
         Nonce
---------       ----                                --
         -----
80398EXAMPLE    Amazon.CodePipeline.Model.JobData   0de392f5-712d-4f41-ace3-
f57a0EXAMPLE    3
```

- Pour API plus de détails, consultez la section PollForJobsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CPJobDetail`

L'exemple de code suivant montre comment utiliser`Get-CPJobDetail`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir des informations générales sur la tâche spécifiée.

```
Get-CPJobDetail -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE
```

Sortie :

```
AccountId       Data                                Id
---------       ----                                --
80398EXAMPLE    Amazon.CodePipeline.Model.JobData
 f570dc12-5ef3-44bc-945a-6e133EXAMPLE
```

Exemple 2 : Cet exemple permet d'obtenir des informations détaillées sur la tâche spécifiée.

```
$jobDetails = Get-CPJobDetail -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE
Write-Output ("For Job " + $jobDetails.Id + ":")
Write-Output ("  AccountId = " + $jobDetails.AccountId)
$jobData = $jobDetails.Data
Write-Output ("  Configuration:")
```

```
ForEach ($key in $jobData.ActionConfiguration.Keys) {
  $value = $jobData.ActionConfiguration.$key
  Write-Output ("    " + $key + " = " + $value)
}
Write-Output ("  ActionTypeId:")
Write-Output ("    Category = " + $jobData.ActionTypeId.Category)
Write-Output ("    Owner = " + $jobData.ActionTypeId.Owner)
Write-Output ("    Provider = " + $jobData.ActionTypeId.Provider)
Write-Output ("    Version = " + $jobData.ActionTypeId.Version)
Write-Output ("  ArtifactCredentials:")
Write-Output ("    AccessKeyId = " + $jobData.ArtifactCredentials.AccessKeyId)
Write-Output ("    SecretAccessKey = " +
 $jobData.ArtifactCredentials.SecretAccessKey)
Write-Output ("    SessionToken = " + $jobData.ArtifactCredentials.SessionToken)
Write-Output ("  InputArtifacts:")
ForEach ($ia in $jobData.InputArtifacts) {
  Write-Output ("    " + $ia.Name)
}
Write-Output ("  OutputArtifacts:")
ForEach ($oa in $jobData.OutputArtifacts) {
  Write-Output ("    " + $oa.Name)
}
Write-Output ("  PipelineContext:")
$context = $jobData.PipelineContext
Write-Output ("    Name = " + $context.Action.Name)
Write-Output ("    PipelineName = " + $context.PipelineName)
Write-Output ("    Stage = " + $context.Stage.Name)
```

Sortie :

```
For Job f570dc12-5ef3-44bc-945a-6e133EXAMPLE:
  AccountId = 80398EXAMPLE
  Configuration:
  ActionTypeId:
    Category = Build
    Owner = Custom
    Provider = MyCustomProviderName
    Version = 1
  ArtifactCredentials:
    AccessKeyId = ASIAIEI3...IXI6YREX
    SecretAccessKey = cqAFDhEi...RdQyfa2u
    SessionToken = AQoDYXdz...5u+lsAU=
  InputArtifacts:
```

```
    MyApp
  OutputArtifacts:
    MyAppBuild
  PipelineContext:
    Name = Build
    PipelineName = CodePipelineDemo
    Stage = Build
```

- Pour API plus de détails, consultez la section GetJobDetailsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CPPipeline`

L'exemple de code suivant montre comment utiliser`Get-CPPipeline`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir des informations générales sur le pipeline spécifié.

```
Get-CPPipeline -Name CodePipelineDemo -Version 1
```

Sortie :

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {Source, Build, Beta, TestStage}
Version       : 1
```

Exemple 2 : Cet exemple permet d'obtenir des informations détaillées sur le pipeline spécifié.

```
$pipeline = Get-CPPipeline -Name CodePipelineDemo
Write-Output ("Name = " + $pipeline.Name)
Write-Output ("RoleArn = " + $pipeline.RoleArn)
Write-Output ("Version = " + $pipeline.Version)
Write-Output ("ArtifactStore:")
Write-Output ("  Location = " + $pipeline.ArtifactStore.Location)
Write-Output ("  Type = " + $pipeline.ArtifactStore.Type.Value)
Write-Output ("Stages:")
ForEach ($stage in $pipeline.Stages) {
  Write-Output ("  Name = " + $stage.Name)
```

```
  Write-Output ("    Actions:")
  ForEach ($action in $stage.Actions) {
    Write-Output ("      Name = " + $action.Name)
 Write-Output ("        Category = " + $action.ActionTypeId.Category)
 Write-Output ("        Owner = " + $action.ActionTypeId.Owner)
 Write-Output ("        Provider = " + $action.ActionTypeId.Provider)
 Write-Output ("        Version = " + $action.ActionTypeId.Version)
 Write-Output ("        Configuration:")
 ForEach ($key in $action.Configuration.Keys) {
   $value = $action.Configuration.$key
   Write-Output ("          " + $key + " = " + $value)
 }
 Write-Output ("        InputArtifacts:")
 ForEach ($ia in $action.InputArtifacts) {
   Write-Output ("          " + $ia.Name)
 }
 ForEach ($oa in $action.OutputArtifacts) {
   Write-Output ("          " + $oa.Name)
 }
 Write-Output ("        RunOrder = " + $action.RunOrder)
  }
}
```

Sortie :

```
Name = CodePipelineDemo
RoleArn = arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Version = 3
ArtifactStore:
  Location = MyBucketName
  Type = S3
Stages:
  Name = Source
    Actions:
      Name = Source
        Category = Source
        Owner = ThirdParty
        Provider = GitHub
        Version = 1
        Configuration:
          Branch = master
          OAuthToken = ****
          Owner = my-user-name
```

```
          Repo = MyRepoName
        InputArtifacts:
          MyApp
        RunOrder = 1
  Name = Build
    Actions:
      Name = Build
        Category = Build
        Owner = Custom
        Provider = MyCustomProviderName
        Version = 1
        Configuration:
          ProjectName = MyProjectName
        InputArtifacts:
          MyApp
          MyAppBuild
        RunOrder = 1
  Name = Beta
    Actions:
      Name = CodePipelineDemoFleet
        Category = Deploy
        Owner = AWS
        Provider = CodeDeploy
        Version = 1
        Configuration:
          ApplicationName = CodePipelineDemoApplication
          DeploymentGroupName = CodePipelineDemoFleet
        InputArtifacts:
          MyAppBuild
        RunOrder = 1
  Name = TestStage
    Actions:
      Name = MyJenkinsTestAction
        Category = Test
        Owner = Custom
        Provider = MyCustomTestProvider
        Version = 1
        Configuration:
          ProjectName = MyJenkinsProjectName
        InputArtifacts:
          MyAppBuild
        RunOrder = 1
```

- Pour API plus de détails, consultez la section GetPipelineRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CPPipelineList`

L'exemple de code suivant montre comment utiliser`Get-CPPipelineList`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir une liste des pipelines disponibles.

```
Get-CPPipelineList
```

Sortie :

```
Created                 Name                Updated                 Version
-------                 ----                -------                 -------
8/13/2015 10:17:54 PM   CodePipelineDemo    8/13/2015 10:17:54 PM   3
7/8/2015 2:41:53 AM     MyFirstPipeline     7/22/2015 9:06:37 PM    7
```

- Pour API plus de détails, consultez la section ListPipelinesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CPPipelineState`

L'exemple de code suivant montre comment utiliser`Get-CPPipelineState`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir des informations générales sur les étapes du pipeline spécifié.

```
Get-CPPipelineState -Name CodePipelineDemo
```

Sortie :

```
Created         : 8/13/2015 10:17:54 PM
PipelineName    : CodePipelineDemo
PipelineVersion : 1
StageStates     : {Source, Build, Beta, TestStage}
```

```
Updated         : 8/13/2015 10:17:54 PM
```

Exemple 2 : Cet exemple permet d'obtenir des informations détaillées sur l'état du pipeline spécifié.

```
ForEach ($stageState in (Get-CPPipelineState -Name $arg).StageStates) {
  Write-Output ("For " + $stageState.StageName + ":")
  Write-Output ("  InboundTransitionState:")
  Write-Output ("    DisabledReason = " +
 $stageState.InboundTransitionState.DisabledReason)
  Write-Output ("    Enabled = " + $stageState.InboundTransitionState.Enabled)
  Write-Output ("    LastChangedAt = " +
 $stageState.InboundTransitionState.LastChangedAt)
  Write-Output ("    LastChangedBy = " +
 $stageState.InboundTransitionState.LastChangedBy)
  Write-Output ("  ActionStates:")
  ForEach ($actionState in $stageState.ActionStates) {
    Write-Output ("    For " + $actionState.ActionName + ":")
 Write-Output ("      CurrentRevision:")
    Write-Output ("        Created = " + $actionState.CurrentRevision.Created)
 Write-Output ("        RevisionChangeId = " +
 $actionState.CurrentRevision.RevisionChangeId)
 Write-Output ("        RevisionId = " + $actionState.CurrentRevision.RevisionId)
 Write-Output ("      EntityUrl = " + $actionState.EntityUrl)
 Write-Output ("      LatestExecution:")
    Write-Output ("        ErrorDetails:")
    Write-Output ("          Code = " +
 $actionState.LatestExecution.ErrorDetails.Code)
 Write-Output ("          Message = " +
 $actionState.LatestExecution.ErrorDetails.Message)
 Write-Output ("        ExternalExecutionId = " +
 $actionState.LatestExecution.ExternalExecutionId)
 Write-Output ("        ExternalExecutionUrl = " +
 $actionState.LatestExecution.ExternalExecutionUrl)
 Write-Output ("        LastStatusChange = " +
 $actionState.LatestExecution.LastStatusChange)
 Write-Output ("        PercentComplete = " +
 $actionState.LatestExecution.PercentComplete)
 Write-Output ("        Status = " + $actionState.LatestExecution.Status)
 Write-Output ("        Summary = " + $actionState.LatestExecution.Summary)
 Write-Output ("      RevisionUrl = " + $actionState.RevisionUrl)
  }
}
```

Sortie :

```
For Source:
  InboundTransitionState:
    DisabledReason =
    Enabled =
    LastChangedAt =
    LastChangedBy =
  ActionStates:
    For Source:
      CurrentRevision:
        Created =
        RevisionChangeId =
        RevisionId =
      EntityUrl = https://github.com/my-user-name/MyRepoName/tree/master
      LatestExecution:
        ErrorDetails:
          Code =
          Message =
        ExternalExecutionId =
        ExternalExecutionUrl =
        LastStatusChange = 07/20/2015 23:28:45
        PercentComplete = 0
        Status = Succeeded
        Summary =
      RevisionUrl =
For Build:
  InboundTransitionState:
    DisabledReason =
    Enabled = True
    LastChangedAt = 01/01/0001 00:00:00
    LastChangedBy =
  ActionStates:
    For Build:
      CurrentRevision:
        Created =
        RevisionChangeId =
        RevisionId =
      EntityUrl = http://54.174.131.1EX/job/MyJenkinsDemo
      LatestExecution:
        ErrorDetails:
          Code = TimeoutError
          Message = The action failed because a job worker exceeded its time limit.
 If this is a custom action, make sure that the job worker is configured correctly.
```

```
        ExternalExecutionId =
        ExternalExecutionUrl =
        LastStatusChange = 07/21/2015 00:29:29
        PercentComplete = 0
        Status = Failed
        Summary =
      RevisionUrl =
For Beta:
  InboundTransitionState:
    DisabledReason =
    Enabled = True
    LastChangedAt = 01/01/0001 00:00:00
    LastChangedBy =
  ActionStates:
    For CodePipelineDemoFleet:
      CurrentRevision:
        Created =
        RevisionChangeId =
        RevisionId =
      EntityUrl = https://console.aws.amazon.com/codedeploy/home?#/applications/
CodePipelineDemoApplication/deployment-groups/CodePipelineDemoFleet
      LatestExecution:
        ErrorDetails:
          Code =
          Message =
        ExternalExecutionId = d-D5LTCZXEX
        ExternalExecutionUrl = https://console.aws.amazon.com/codedeploy/home?#/
deployments/d-D5LTCZXEX
        LastStatusChange = 07/08/2015 22:07:42
        PercentComplete = 0
        Status = Succeeded
        Summary = Deployment Succeeded
      RevisionUrl =
For TestStage:
  InboundTransitionState:
    DisabledReason =
    Enabled = True
    LastChangedAt = 01/01/0001 00:00:00
    LastChangedBy =
  ActionStates:
    For MyJenkinsTestAction25:
      CurrentRevision:
        Created =
        RevisionChangeId =
```

```
        RevisionId =
      EntityUrl = http://54.174.131.1EX/job/MyJenkinsDemo
      LatestExecution:
        ErrorDetails:
          Code =
          Message =
        ExternalExecutionId = 5
        ExternalExecutionUrl = http://54.174.131.1EX/job/MyJenkinsDemo/5
        LastStatusChange = 07/08/2015 22:09:03
        PercentComplete = 0
        Status = Succeeded
        Summary = Finished
      RevisionUrl =
```

- Pour API plus de détails, consultez la section GetPipelineStateRéférence des AWS Tools for PowerShell applets de commande.

### `New-CPCustomActionType`

L'exemple de code suivant montre comment utiliser`New-CPCustomActionType`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une nouvelle action personnalisée avec les propriétés spécifiées.

```
New-CPCustomActionType -Category Build -ConfigurationProperty @{"Description"
 = "The name of the build project must be provided when this action is added
 to the pipeline."; "Key" = $True; "Name" = "ProjectName"; "Queryable"
 = $False; "Required" = $True; "Secret" = $False; "Type" = "String"} -
Settings_EntityUrlTemplate "https://my-build-instance/job/{Config:ProjectName}/"
 -Settings_ExecutionUrlTemplate "https://my-build-instance/job/mybuildjob/
lastSuccessfulBuild{ExternalExecutionId}/" -InputArtifactDetails_MaximumCount
 1 -OutputArtifactDetails_MaximumCount 1 -InputArtifactDetails_MinimumCount 0 -
OutputArtifactDetails_MinimumCount 0 -Provider "MyBuildProviderName" -Version 1
```

Sortie :

```
ActionConfigurationProperties : {ProjectName}
Id                            : Amazon.CodePipeline.Model.ActionTypeId
InputArtifactDetails          : Amazon.CodePipeline.Model.ArtifactDetails
OutputArtifactDetails         : Amazon.CodePipeline.Model.ArtifactDetails
Settings                      : Amazon.CodePipeline.Model.ActionTypeSettings
```

- Pour API plus de détails, consultez la section CreateCustomActionTypeRéférence des AWS Tools for PowerShell applets de commande.

### `New-CPPipeline`

L'exemple de code suivant montre comment utiliser`New-CPPipeline`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un nouveau pipeline avec les paramètres spécifiés.

```
$pipeline = New-Object Amazon.CodePipeline.Model.PipelineDeclaration

$sourceStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration
$deployStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration

$sourceStageActionOutputArtifact = New-Object
 Amazon.CodePipeline.Model.OutputArtifact
$sourceStageActionOutputArtifact.Name = "MyApp"

$sourceStageAction.ActionTypeId = @{"Category" = "Source"; "Owner" = "AWS";
 "Provider" = "S3"; "Version" = 1}
$sourceStageAction.Configuration.Add("S3Bucket", "amzn-s3-demo-bucket")
$sourceStageAction.Configuration.Add("S3ObjectKey", "my-object-key-name.zip")
$sourceStageAction.OutputArtifacts.Add($sourceStageActionOutputArtifact)
$sourceStageAction.Name = "Source"

$deployStageActionInputArtifact = New-Object Amazon.CodePipeline.Model.InputArtifact
$deployStageActionInputArtifact.Name = "MyApp"

$deployStageAction.ActionTypeId = @{"Category" = "Deploy"; "Owner" = "AWS";
 "Provider" = "CodeDeploy"; "Version" = 1}
$deployStageAction.Configuration.Add("ApplicationName",
 "CodePipelineDemoApplication")
$deployStageAction.Configuration.Add("DeploymentGroupName", "CodePipelineDemoFleet")
$deployStageAction.InputArtifacts.Add($deployStageActionInputArtifact)
$deployStageAction.Name = "CodePipelineDemoFleet"

$sourceStage = New-Object Amazon.CodePipeline.Model.StageDeclaration
$deployStage = New-Object Amazon.CodePipeline.Model.StageDeclaration

$sourceStage.Name = "Source"
$deployStage.Name = "Beta"
```

```
$sourceStage.Actions.Add($sourceStageAction)
$deployStage.Actions.Add($deployStageAction)

$pipeline.ArtifactStore = @{"Location" = "amzn-s3-demo-bucket"; "Type" = "S3"}
$pipeline.Name = "CodePipelineDemo"
$pipeline.RoleArn = "arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole"
$pipeline.Stages.Add($sourceStage)
$pipeline.Stages.Add($deployStage)
$pipeline.Version = 1

New-CPPipeline -Pipeline $pipeline
```

Sortie :

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {Source, Beta}
Version       : 1
```

- Pour API plus de détails, consultez la section CreatePipelineRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-CPCustomActionType`

L'exemple de code suivant montre comment utiliser`Remove-CPCustomActionType`.

Outils pour PowerShell

Exemple 1 : cet exemple supprime l'action personnalisée spécifiée. La commande vous demandera une confirmation avant de continuer. Ajoutez le paramètre -Force pour supprimer l'action personnalisée sans invite.

```
Remove-CPCustomActionType -Category Build -Provider MyBuildProviderName -Version 1
```

- Pour API plus de détails, consultez la section DeleteCustomActionTypeRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-CPPipeline`

L'exemple de code suivant montre comment utiliser`Remove-CPPipeline`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le pipeline spécifié. La commande vous demandera une confirmation avant de continuer. Ajoutez le paramètre -Force pour supprimer le pipeline sans y être invité.

```
Remove-CPPipeline -Name CodePipelineDemo
```

- Pour API plus de détails, consultez la section DeletePipelineRéférence des AWS Tools for PowerShell applets de commande.

### `Start-CPPipelineExecution`

L'exemple de code suivant montre comment utiliser`Start-CPPipelineExecution`.

Outils pour PowerShell

Exemple 1 : Cet exemple commence à exécuter le pipeline spécifié.

```
Start-CPPipelineExecution -Name CodePipelineDemo
```

- Pour API plus de détails, consultez la section StartPipelineExecutionRéférence des AWS Tools for PowerShell applets de commande.

### `Update-CPPipeline`

L'exemple de code suivant montre comment utiliser`Update-CPPipeline`.

Outils pour PowerShell

Exemple 1 : Cet exemple met à jour le pipeline existant spécifié avec les paramètres spécifiés.

```
$pipeline = New-Object Amazon.CodePipeline.Model.PipelineDeclaration

$sourceStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration
$deployStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration
```

```
$sourceStageActionOutputArtifact = New-Object
 Amazon.CodePipeline.Model.OutputArtifact
$sourceStageActionOutputArtifact.Name = "MyApp"

$sourceStageAction.ActionTypeId = @{"Category" = "Source"; "Owner" = "AWS";
 "Provider" = "S3"; "Version" = 1}
$sourceStageAction.Configuration.Add("S3Bucket", "amzn-s3-demo-bucket")
$sourceStageAction.Configuration.Add("S3ObjectKey", "my-object-key-name.zip")
$sourceStageAction.OutputArtifacts.Add($sourceStageActionOutputArtifact)
$sourceStageAction.Name = "Source"

$deployStageActionInputArtifact = New-Object Amazon.CodePipeline.Model.InputArtifact
$deployStageActionInputArtifact.Name = "MyApp"

$deployStageAction.ActionTypeId = @{"Category" = "Deploy"; "Owner" = "AWS";
 "Provider" = "CodeDeploy"; "Version" = 1}
$deployStageAction.Configuration.Add("ApplicationName",
 "CodePipelineDemoApplication")
$deployStageAction.Configuration.Add("DeploymentGroupName", "CodePipelineDemoFleet")
$deployStageAction.InputArtifacts.Add($deployStageActionInputArtifact)
$deployStageAction.Name = "CodePipelineDemoFleet"

$sourceStage = New-Object Amazon.CodePipeline.Model.StageDeclaration
$deployStage = New-Object Amazon.CodePipeline.Model.StageDeclaration

$sourceStage.Name = "MyInputFiles"
$deployStage.Name = "MyTestDeployment"

$sourceStage.Actions.Add($sourceStageAction)
$deployStage.Actions.Add($deployStageAction)

$pipeline.ArtifactStore = @{"Location" = "amzn-s3-demo-bucket"; "Type" = "S3"}
$pipeline.Name = "CodePipelineDemo"
$pipeline.RoleArn = "arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole"
$pipeline.Stages.Add($sourceStage)
$pipeline.Stages.Add($deployStage)
$pipeline.Version = 1

Update-CPPipeline -Pipeline $pipeline
```

Sortie :

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
```

```
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {InputFiles, TestDeployment}
Version       : 2
```

- Pour API plus de détails, consultez la section UpdatePipelineRéférence des AWS Tools for PowerShell applets de commande.

# Exemples d'Amazon Cognito Identity utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS Tools for PowerShell aide d'Amazon Cognito Identity.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `Get-CGIIdentityPool`

L'exemple de code suivant montre comment utiliser`Get-CGIIdentityPool`.

Outils pour PowerShell

Exemple 1 : récupère les informations relatives à un pool d'identités spécifique par son identifiant.

```
Get-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
```

Sortie :

```
LoggedAt                      : 8/12/2015 4:29:40 PM
```

```
AllowUnauthenticatedIdentities : True
DeveloperProviderName          :
IdentityPoolId                 : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
IdentityPoolName               : CommonTests1
OpenIdConnectProviderARNs      : {}
SupportedLoginProviders        : {}
ResponseMetadata               : Amazon.Runtime.ResponseMetadata
ContentLength                  : 142
HttpStatusCode                 : OK
```

- Pour API plus de détails, consultez la section DescribeIdentityPoolRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CGIIdentityPoolList`

L'exemple de code suivant montre comment utiliser`Get-CGIIdentityPoolList`.

Outils pour PowerShell

Exemple 1 : récupère une liste de pools d'identités existants.

```
Get-CGIIdentityPoolList
```

Sortie :

```
IdentityPoolId                                                  IdentityPoolName
--------------                                                  ----------------
us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1                  CommonTests1
us-east-1:118d242d-204e-4b88-b803-EXAMPLEGUID2                  Tests2
us-east-1:15d49393-ab16-431a-b26e-EXAMPLEGUID3                  CommonTests13
```

- Pour API plus de détails, consultez la section ListIdentityPoolsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CGIIdentityPoolRole`

L'exemple de code suivant montre comment utiliser`Get-CGIIdentityPoolRole`.

Outils pour PowerShell

Exemple 1 : obtient les informations sur les rôles pour un pool d'identités spécifique.

```
Get-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

Sortie :

```
LoggedAt         : 8/12/2015 4:33:51 PM
IdentityPoolId   : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
Roles            : {[unauthenticated, arn:aws:iam::123456789012:role/
CommonTests1Role]}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 165
HttpStatusCode   : OK
```

- Pour API plus de détails, consultez la section GetIdentityPoolRolesRéférence des AWS Tools for PowerShell applets de commande.

### `New-CGIIdentityPool`

L'exemple de code suivant montre comment utiliser`New-CGIIdentityPool`.

Outils pour PowerShell

Exemple 1 : crée un nouveau pool d'identités qui autorise les identités non authentifiées.

```
New-CGIIdentityPool -AllowUnauthenticatedIdentities $true -IdentityPoolName
 CommonTests13
```

Sortie :

```
LoggedAt                       : 8/12/2015 4:56:07 PM
AllowUnauthenticatedIdentities : True
DeveloperProviderName          :
IdentityPoolId                 : us-east-1:15d49393-ab16-431a-b26e-EXAMPLEGUID3
IdentityPoolName               : CommonTests13
OpenIdConnectProviderARNs      : {}
SupportedLoginProviders        : {}
ResponseMetadata               : Amazon.Runtime.ResponseMetadata
ContentLength                  : 136
HttpStatusCode                 : OK
```

- Pour API plus de détails, consultez la section CreateIdentityPoolRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-CGIIdentityPool`

L'exemple de code suivant montre comment utiliser`Remove-CGIIdentityPool`.

Outils pour PowerShell

Exemple 1 : Supprime un pool d'identités spécifique.

```
Remove-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

- Pour API plus de détails, consultez la section DeleteIdentityPoolRéférence des AWS Tools for PowerShell applets de commande.

### `Set-CGIIdentityPoolRole`

L'exemple de code suivant montre comment utiliser`Set-CGIIdentityPoolRole`.

Outils pour PowerShell

Exemple 1 : configure le pool d'identités spécifique pour qu'il ait un rôle non authentifiéIAM.

```
Set-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -Role @{ "unauthenticated" = "arn:aws:iam::123456789012:role/
CommonTests1Role" }
```

- Pour API plus de détails, consultez la section SetIdentityPoolRolesRéférence des AWS Tools for PowerShell applets de commande.

### `Update-CGIIdentityPool`

L'exemple de code suivant montre comment utiliser`Update-CGIIdentityPool`.

Outils pour PowerShell

Exemple 1 : met à jour certaines propriétés du pool d'identités, en l'occurrence le nom du pool d'identités.

```
Update-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -IdentityPoolName NewPoolName
```

Sortie :

```
LoggedAt                       : 8/12/2015 4:53:33 PM
AllowUnauthenticatedIdentities : False
DeveloperProviderName          :
IdentityPoolId                 : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
IdentityPoolName               : NewPoolName
OpenIdConnectProviderARNs      : {}
SupportedLoginProviders        : {}
ResponseMetadata               : Amazon.Runtime.ResponseMetadata
ContentLength                  : 135
HttpStatusCode                 : OK
```

- Pour API plus de détails, consultez la section UpdateIdentityPoolRéférence des AWS Tools for PowerShell applets de commande.

# AWS Config exemples utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS Tools for PowerShell with AWS Config.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `Add-CFGResourceTag`

L'exemple de code suivant montre comment utiliser`Add-CFGResourceTag`.

Outils pour PowerShell

Exemple 1 : Cet exemple associe la balise spécifiée à la ressourceARN, qui est config-rule/config-rule-16iyn0 dans ce cas.

```
Add-CFGResourceTag -ResourceArn arn:aws:config:eu-west-1:123456789012:config-rule/
config-rule-16iyn0 -Tag @{Key="Release";Value="Beta"}
```

- Pour API plus de détails, consultez la section TagResourceRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFGAggregateComplianceByConfigRuleList`

L'exemple de code suivant montre comment utiliser`Get-CFGAggregateComplianceByConfigRuleList`.

Outils pour PowerShell

Exemple 1 : Cet exemple extrait les détails du filtrage ConfigurationAggregator « kaju » pour la règle de configuration donnée et développe/renvoie le niveau de « conformité » de la règle.

```
Get-CFGAggregateComplianceByConfigRuleList -ConfigurationAggregatorName kaju
 -Filters_ConfigRuleName ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK | Select-Object -
ExpandProperty Compliance
```

Sortie :

```
ComplianceContributorCount                          ComplianceType
--------------------------                          --------------
Amazon.ConfigService.Model.ComplianceContributorCount NON_COMPLIANT
```

Exemple 2 : Cet exemple extrait les détails du compte donné ConfigurationAggregator, les filtre pour le compte donné pour toutes les régions couvertes par l'agrégateur et indique ensuite la conformité de toutes les règles.

```
Get-CFGAggregateComplianceByConfigRuleList -ConfigurationAggregatorName
 kaju -Filters_AccountId 123456789012 | Select-Object ConfigRuleName,
 @{N="Compliance";E={$_.Compliance.ComplianceType}}
```

Sortie :

```
ConfigRuleName                      Compliance
--------------                      ----------
ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK NON_COMPLIANT
ec2-instance-no-public-ip           NON_COMPLIANT
desired-instance-type               NON_COMPLIANT
```

- Pour API plus de détails, consultez la section DescribeAggregateComplianceByConfigRulesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFGAggregateComplianceDetailsByConfigRule`

L'exemple de code suivant montre comment utiliser`Get-CFGAggregateComplianceDetailsByConfigRule`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie les résultats de l'évaluation en sélectionnant la sortie avec l'identifiant de ressource et le type de ressource pour la règle de AWS configuration « » qui sont à l'état desired-instance-type « » pour le compte, l'agrégateur, la région et la règle de configuration donnés COMPLIANT

```
Get-CFGAggregateComplianceDetailsByConfigRule -AccountId 123456789012 -
AwsRegion eu-west-1 -ComplianceType COMPLIANT -ConfigRuleName desired-
instance-type -ConfigurationAggregatorName raju | Select-Object -
ExpandProperty EvaluationResultIdentifier | Select-Object -ExpandProperty
 EvaluationResultQualifier
```

Sortie :

```
ConfigRuleName        ResourceId          ResourceType
--------------        ----------          ------------
desired-instance-type i-0f1bf2f34c5678d12 AWS::EC2::Instance
desired-instance-type i-0fd12dd3456789123 AWS::EC2::Instance
```

- Pour API plus de détails, consultez la section GetAggregateComplianceDetailsByConfigRuleRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFGAggregateConfigRuleComplianceSummary`

L'exemple de code suivant montre comment utiliser`Get-CFGAggregateConfigRuleComplianceSummary`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie le nombre de règles non conformes pour l'agrégateur donné.

```
(Get-CFGAggregateConfigRuleComplianceSummary -ConfigurationAggregatorName
 raju).AggregateComplianceCounts.ComplianceSummary.NonCompliantResourceCount
```

Sortie :

```
CapExceeded CappedCount
----------- -----------
False       5
```

- Pour API plus de détails, consultez la section [GetAggregateConfigRuleComplianceSummary](#)Référence des AWS Tools for PowerShell applets de commande.

### `Get-CFGAggregateDiscoveredResourceCount`

L'exemple de code suivant montre comment utiliser`Get-CFGAggregateDiscoveredResourceCount`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie le nombre de ressources pour l'agrégateur donné filtré pour la région us-east-1.

```
Get-CFGAggregateDiscoveredResourceCount -ConfigurationAggregatorName Master -
Filters_Region us-east-1
```

Sortie :

```
GroupByKey GroupedResourceCounts NextToken TotalDiscoveredResources
---------- -------------------- --------- -----------------------
```

```
            {}                                  455
```

Exemple 2 : Cet exemple renvoie le nombre de ressources groupé par RESOURCE _ TYPE pour la région filtrée pour l'agrégateur donné.

```
Get-CFGAggregateDiscoveredResourceCount -ConfigurationAggregatorName Master -
Filters_Region us-east-1 -GroupByKey RESOURCE_TYPE |
   Select-Object -ExpandProperty GroupedResourceCounts
```

Sortie :

```
GroupName                          ResourceCount
---------                          -------------
AWS::CloudFormation::Stack         12
AWS::CloudFront::Distribution      1
AWS::CloudTrail::Trail             1
AWS::DynamoDB::Table               1
AWS::EC2::EIP                      2
AWS::EC2::FlowLog                  2
AWS::EC2::InternetGateway          4
AWS::EC2::NatGateway               2
AWS::EC2::NetworkAcl               4
AWS::EC2::NetworkInterface         12
AWS::EC2::RouteTable               13
AWS::EC2::SecurityGroup            18
AWS::EC2::Subnet                   16
AWS::EC2::VPC                      4
AWS::EC2::VPCEndpoint              2
AWS::EC2::VPCPeeringConnection     1
AWS::IAM::Group                    2
AWS::IAM::Policy                   51
AWS::IAM::Role                     78
AWS::IAM::User                     7
AWS::Lambda::Function              3
AWS::RDS::DBSecurityGroup          1
AWS::S3::Bucket                    3
AWS::SSM::AssociationCompliance    107
AWS::SSM::ManagedInstanceInventory 108
```

- Pour API plus de détails, consultez la section GetAggregateDiscoveredResourceCountsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFGAggregateDiscoveredResourceList`

L'exemple de code suivant montre comment utiliser`Get-CFGAggregateDiscoveredResourceList`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie les identifiants de ressources pour le type de ressource donné agrégés dans l'agrégateur « Irlande ». Pour la liste des types de ressources, veuillez consulter le https://docs.aws.amazon.com/sdkfornet/ fichier v3/apidocs/index.html ? page=ConfigService/TConfigServiceResourceTypeConfigService.html&tocid=Amazon_ _. ResourceType

```
Get-CFGAggregateDiscoveredResourceList -ConfigurationAggregatorName Ireland -
ResourceType ([Amazon.ConfigService.ResourceType]::AWSAutoScalingAutoScalingGroup)
```

Sortie :

```
ResourceId      : arn:aws:autoscaling:eu-
west-1:123456789012:autoScalingGroup:12e3b4fc-1234-1234-
a123-1d2ba3c45678:autoScalingGroupName/asg-1
ResourceName    : asg-1
ResourceType    : AWS::AutoScaling::AutoScalingGroup
SourceAccountId : 123456789012
SourceRegion    : eu-west-1
```

Exemple 2 : Cet exemple renvoie le type de ressource **`AwsEC2SecurityGroup`** nommé « default » pour l'agrégateur donné filtré avec la région us-east-1.

```
Get-CFGAggregateDiscoveredResourceList -ConfigurationAggregatorName raju -
ResourceType ([Amazon.ConfigService.ResourceType]::AWSEC2SecurityGroup) -
Filters_Region us-east-1 -Filters_ResourceName default
```

Sortie :

```
ResourceId      : sg-01234bd5dbfa67c89
ResourceName    : default
ResourceType    : AWS::EC2::SecurityGroup
SourceAccountId : 123456789102
SourceRegion    : us-east-1

ResourceId      : sg-0123a4ebbf56789be
```

```
ResourceName    : default
ResourceType    : AWS::EC2::SecurityGroup
SourceAccountId : 123456789102
SourceRegion    : us-east-1

ResourceId      : sg-4fc1d234
ResourceName    : default
ResourceType    : AWS::EC2::SecurityGroup
SourceAccountId : 123456789102
SourceRegion    : us-east-1
```

- Pour API plus de détails, consultez la section ListAggregateDiscoveredResourcesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFGAggregateResourceConfig`

L'exemple de code suivant montre comment utiliser`Get-CFGAggregateResourceConfig`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie l'élément de configuration pour la ressource agrégée donnée et développe la configuration.

```
(Get-CFGAggregateResourceConfig -ResourceIdentifier_SourceRegion
 us-east-1 -ResourceIdentifier_SourceAccountId 123456789012 -
ResourceIdentifier_ResourceId sg-4fc1d234 -ResourceIdentifier_ResourceType
 ([Amazon.ConfigService.ResourceType]::AWSEC2SecurityGroup) -
ConfigurationAggregatorName raju).Configuration | ConvertFrom-Json
```

Sortie :

```
{"description":"default VPC security group","groupName":"default","ipPermissions":
[{"ipProtocol":"-1","ipv6Ranges":[],"prefixListIds":[],"userIdGroupPairs":
[{"groupId":"sg-4fc1d234","userId":"123456789012"}],"ipv4Ranges":
[],"ipRanges":[]},{"fromPort":3389,"ipProtocol":"tcp","ipv6Ranges":
[],"prefixListIds":[],"toPort":3389,"userIdGroupPairs":[],"ipv4Ranges":
[{"cidrIp":"54.240.197.224/29","description":"office subnet"},
{"cidrIp":"72.21.198.65/32","description":"home pc"}],"ipRanges":
["54.240.197.224/29","72.21.198.65/32"]}],"ownerId":"123456789012","groupId":"sg-4fc1d234","
[{"ipProtocol":"-1","ipv6Ranges":[],"prefixListIds":[],"userIdGroupPairs":
[],"ipv4Ranges":[{"cidrIp":"0.0.0.0/0"}],"ipRanges":["0.0.0.0/0"]}],"tags":
[],"vpcId":"vpc-2d1c2e34"}
```

- Pour API plus de détails, consultez GetAggregateResourceconfig-service dans le manuel AWS Tools for PowerShell Cmdlet Reference.

### `Get-CFGAggregateResourceConfigBatch`

L'exemple de code suivant montre comment utiliser`Get-CFGAggregateResourceConfigBatch`.

Outils pour PowerShell

Exemple 1 : Cet exemple récupère l'élément de configuration actuel pour la ressource (identifiée) présente dans l'agrégateur donné.

```
$resIdentifier=[Amazon.ConfigService.Model.AggregateResourceIdentifier]@{
  ResourceId= "i-012e3cb4df567e8aa"
  ResourceName = "arn:aws:ec2:eu-west-1:123456789012:instance/i-012e3cb4df567e8aa"
  ResourceType = [Amazon.ConfigService.ResourceType]::AWSEC2Instance
  SourceAccountId = "123456789012"
  SourceRegion = "eu-west-1"
 }

 Get-CFGAggregateResourceConfigBatch -ResourceIdentifier $resIdentifier -
ConfigurationAggregatorName raju
```

Sortie :

```
BaseConfigurationItems UnprocessedResourceIdentifiers
---------------------- ------------------------------
{}                     {arn:aws:ec2:eu-west-1:123456789012:instance/
i-012e3cb4df567e8aa}
```

- Pour API plus de détails, consultez BatchGetAggregateResourceconfig-service dans le manuel AWS Tools for PowerShell Cmdlet Reference.

### `Get-CFGAggregationAuthorizationList`

L'exemple de code suivant montre comment utiliser`Get-CFGAggregationAuthorizationList`.

Outils pour PowerShell

Exemple 1 : Cet exemple récupère les autorisations accordées aux agrégateurs.

```
Get-CFGAggregationAuthorizationList
```

Sortie :

```
AggregationAuthorizationArn
   AuthorizedAccountId AuthorizedAwsRegion CreationTime
--------------------------
   ------------------ ------------------ ------------
arn:aws:config-service:eu-west-1:123456789012:aggregation-
authorization/123456789012/eu-west-1 123456789012        eu-west-1
 8/26/2019 12:55:27 AM
```

- Pour API plus de détails, consultez la section DescribeAggregationAuthorizationsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFGComplianceByConfigRule`

L'exemple de code suivant montre comment utiliser`Get-CFGComplianceByConfigRule`.

Outils pour PowerShell

Exemple 1 : Cet exemple récupère les détails de conformité de la règle ebs-optimized-instance, pour laquelle il n'existe aucun résultat d'évaluation actuel pour la règle, il renvoie donc _ INSUFFICIENT DATA

```
(Get-CFGComplianceByConfigRule -ConfigRuleName ebs-optimized-instance).Compliance
```

Sortie :

```
ComplianceContributorCount ComplianceType
-------------------------- --------------
                           INSUFFICIENT_DATA
```

Exemple 2 : Cet exemple renvoie le nombre de ressources non conformes pour la règle ALB _ HTTP _TO_ _ _HTTPS. REDIRECTION CHECK

```
(Get-CFGComplianceByConfigRule -ConfigRuleName ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK -
ComplianceType NON_COMPLIANT).Compliance.ComplianceContributorCount
```

Sortie :

```
CapExceeded CappedCount
----------- -----------
False       2
```

- Pour API plus de détails, consultez la section DescribeComplianceByConfigRuleRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFGComplianceByResource`

L'exemple de code suivant montre comment utiliser`Get-CFGComplianceByResource`.

Outils pour PowerShell

Exemple 1 : Cet exemple vérifie le type de **`AWS::SSM::ManagedInstanceInventory`** ressource pour le type de conformité COMPLIANT « ».

```
Get-CFGComplianceByResource -ComplianceType COMPLIANT -ResourceType
 AWS::SSM::ManagedInstanceInventory
```

Sortie :

```
Compliance                              ResourceId          ResourceType
----------                              ----------          ------------
Amazon.ConfigService.Model.Compliance i-0123bcf4b567890e3
 AWS::SSM::ManagedInstanceInventory
Amazon.ConfigService.Model.Compliance i-0a1234f6f5d6b78f7
 AWS::SSM::ManagedInstanceInventory
```

- Pour API plus de détails, consultez la section DescribeComplianceByResourceRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFGComplianceDetailsByConfigRule`

L'exemple de code suivant montre comment utiliser`Get-CFGComplianceDetailsByConfigRule`.

Outils pour PowerShell

Exemple 1 : Cet exemple obtient les résultats de l'évaluation de la règle access-keys-rotated et renvoie la sortie groupée par type de conformité

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-keys-rotated | Group-
Object ComplianceType
```

Sortie :

```
Count Name                      Group
----- ----                      -----
    2 COMPLIANT                 {Amazon.ConfigService.Model.EvaluationResult,
 Amazon.ConfigService.Model.EvaluationResult}
    5 NON_COMPLIANT             {Amazon.ConfigService.Model.EvaluationResult,
 Amazon.ConfigService.Model.EvaluationResult,
 Amazon.ConfigService.Model.EvaluationRes...
```

Exemple 2 : Cet exemple interroge les détails de conformité de la règle access-keys-rotated relative aux COMPLIANT ressources.

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-
keys-rotated -ComplianceType COMPLIANT | ForEach-Object
 {$_.EvaluationResultIdentifier.EvaluationResultQualifier}
```

Sortie :

```
ConfigRuleName      ResourceId            ResourceType
--------------      ----------            ------------
access-keys-rotated BCAB1CDJ2LITAPVEW3JAH AWS::IAM::User
access-keys-rotated BCAB1CDJ2LITL3EHREM4Q AWS::IAM::User
```

- Pour API plus de détails, consultez la section GetComplianceDetailsByConfigRuleRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFGComplianceDetailsByResource`

L'exemple de code suivant montre comment utiliser`Get-CFGComplianceDetailsByResource`.

Outils pour PowerShell

Exemple 1 : Cet exemple d'évaluation donne des résultats pour la ressource donnée.

```
Get-CFGComplianceDetailsByResource -ResourceId ABCD5STJ4EFGHIVEW6JAH -ResourceType
 'AWS::IAM::User'
```

Sortie :

```
Annotation                :
ComplianceType            : COMPLIANT
ConfigRuleInvokedTime     : 8/25/2019 11:34:56 PM
EvaluationResultIdentifier : Amazon.ConfigService.Model.EvaluationResultIdentifier
ResultRecordedTime        : 8/25/2019 11:34:56 PM
ResultToken               :
```

- Pour API plus de détails, consultez la section GetComplianceDetailsByResourceRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFGComplianceSummaryByConfigRule`

L'exemple de code suivant montre comment utiliser`Get-CFGComplianceSummaryByConfigRule`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie le nombre de règles Config non conformes.

```
Get-CFGComplianceSummaryByConfigRule -Select
 ComplianceSummary.NonCompliantResourceCount
```

Sortie :

```
CapExceeded CappedCount
----------- -----------
False       9
```

- Pour API plus de détails, consultez la section GetComplianceSummaryByConfigRuleRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFGComplianceSummaryByResourceType`

L'exemple de code suivant montre comment utiliser`Get-CFGComplianceSummaryByResourceType`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie le nombre de ressources conformes ou non conformes et convertit la sortie en json.

```
Get-CFGComplianceSummaryByResourceType -Select
 ComplianceSummariesByResourceType.ComplianceSummary | ConvertTo-Json
{
  "ComplianceSummaryTimestamp": "2019-12-14T06:14:49.778Z",
  "CompliantResourceCount": {
    "CapExceeded": false,
    "CappedCount": 2
  },
  "NonCompliantResourceCount": {
    "CapExceeded": true,
    "CappedCount": 100
  }
}
```

- Pour API plus de détails, consultez la section GetComplianceSummaryByResourceTypeRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFGConfigRule`

L'exemple de code suivant montre comment utiliser`Get-CFGConfigRule`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie les règles de configuration du compte, avec les propriétés sélectionnées.

```
Get-CFGConfigRule | Select-Object ConfigRuleName, ConfigRuleId, ConfigRuleArn,
 ConfigRuleState
```

Sortie :

```
ConfigRuleName                                  ConfigRuleId       ConfigRuleArn
                                                    ConfigRuleState
--------------                                  ------------       -------------
                                                    ---------------
ALB_REDIRECTION_CHECK                           config-rule-12iyn3 arn:aws:config-
service:eu-west-1:123456789012:config-rule/config-rule-12iyn3 ACTIVE
access-keys-rotated                             config-rule-aospfr arn:aws:config-
service:eu-west-1:123456789012:config-rule/config-rule-aospfr ACTIVE
```

```
autoscaling-group-elb-healthcheck-required       config-rule-cn1f2x arn:aws:config-
service:eu-west-1:123456789012:config-rule/config-rule-cn1f2x ACTIVE
```

- Pour API plus de détails, consultez la section DescribeConfigRulesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFGConfigRuleEvaluationStatus`

L'exemple de code suivant montre comment utiliser`Get-CFGConfigRuleEvaluationStatus`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie les informations d'état pour les règles de configuration données.

```
Get-CFGConfigRuleEvaluationStatus -ConfigRuleName root-account-mfa-enabled, vpc-
flow-logs-enabled
```

Sortie :

```
ConfigRuleArn                : arn:aws:config:eu-west-1:123456789012:config-rule/
config-rule-kvq1wk
ConfigRuleId                 : config-rule-kvq1wk
ConfigRuleName               : root-account-mfa-enabled
FirstActivatedTime           : 8/27/2019 8:05:17 AM
FirstEvaluationStarted       : True
LastErrorCode                :
LastErrorMessage             :
LastFailedEvaluationTime     : 1/1/0001 12:00:00 AM
LastFailedInvocationTime     : 1/1/0001 12:00:00 AM
LastSuccessfulEvaluationTime : 12/13/2019 8:12:03 AM
LastSuccessfulInvocationTime : 12/13/2019 8:12:03 AM

ConfigRuleArn                : arn:aws:config:eu-west-1:123456789012:config-rule/
config-rule-z1s23b
ConfigRuleId                 : config-rule-z1s23b
ConfigRuleName               : vpc-flow-logs-enabled
FirstActivatedTime           : 8/14/2019 6:23:44 AM
FirstEvaluationStarted       : True
LastErrorCode                :
LastErrorMessage             :
LastFailedEvaluationTime     : 1/1/0001 12:00:00 AM
LastFailedInvocationTime     : 1/1/0001 12:00:00 AM
```

```
LastSuccessfulEvaluationTime : 12/13/2019 7:12:01 AM
LastSuccessfulInvocationTime : 12/13/2019 7:12:01 AM
```

- Pour API plus de détails, consultez la section DescribeConfigRuleEvaluationStatusRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFGConfigurationAggregatorList`

L'exemple de code suivant montre comment utiliser`Get-CFGConfigurationAggregatorList`.

Outils pour PowerShell

Exemple 1 : Cet échantillon renvoie tous les agrégateurs de la région/du compte.

```
Get-CFGConfigurationAggregatorList
```

Sortie :

```
AccountAggregationSources     :
 {Amazon.ConfigService.Model.AccountAggregationSource}
ConfigurationAggregatorArn    : arn:aws:config-service:eu-
west-1:123456789012:config-aggregator/config-aggregator-xabca1me
ConfigurationAggregatorName   : IrelandMaster
CreationTime                  : 8/25/2019 11:42:39 PM
LastUpdatedTime               : 8/25/2019 11:42:39 PM
OrganizationAggregationSource :

AccountAggregationSources     : {}
ConfigurationAggregatorArn    : arn:aws:config-service:eu-
west-1:123456789012:config-aggregator/config-aggregator-qubqabcd
ConfigurationAggregatorName   : raju
CreationTime                  : 8/11/2019 8:39:25 AM
LastUpdatedTime               : 8/11/2019 8:39:25 AM
OrganizationAggregationSource :
 Amazon.ConfigService.Model.OrganizationAggregationSource
```

- Pour API plus de détails, consultez la section DescribeConfigurationAggregatorsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFGConfigurationAggregatorSourcesStatus`

L'exemple de code suivant montre comment utiliser`Get-CFGConfigurationAggregatorSourcesStatus`.

Outils pour PowerShell

Exemple 1 : Cet exemple affiche les champs demandés pour les sources dans l'agrégateur donné.

```
Get-CFGConfigurationAggregatorSourcesStatus -ConfigurationAggregatorName raju |
 select SourceType, LastUpdateStatus, LastUpdateTime, SourceId
```

Sortie :

```
SourceType   LastUpdateStatus LastUpdateTime        SourceId
----------   ---------------- --------------        --------
ORGANIZATION SUCCEEDED        12/31/2019 7:45:06 AM Organization
ACCOUNT      SUCCEEDED        12/31/2019 7:09:38 AM 612641234567
ACCOUNT      SUCCEEDED        12/31/2019 7:12:53 AM 933301234567
ACCOUNT      SUCCEEDED        12/31/2019 7:18:10 AM 933301234567
ACCOUNT      SUCCEEDED        12/31/2019 7:25:17 AM 933301234567
ACCOUNT      SUCCEEDED        12/31/2019 7:25:49 AM 612641234567
ACCOUNT      SUCCEEDED        12/31/2019 7:26:11 AM 612641234567
```

- Pour API plus de détails, consultez la section DescribeConfigurationAggregatorSourcesStatusRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFGConfigurationRecorder`

L'exemple de code suivant montre comment utiliser`Get-CFGConfigurationRecorder`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie les détails des enregistreurs de configuration.

```
Get-CFGConfigurationRecorder | Format-List
```

Sortie :

```
Name           : default
RecordingGroup : Amazon.ConfigService.Model.RecordingGroup
RoleARN        : arn:aws:iam::123456789012:role/aws-service-role/
config.amazonaws.com/AWSServiceRoleForConfig
```

- Pour API plus de détails, consultez la section DescribeConfigurationRecordersRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFGConfigurationRecorderStatus`

L'exemple de code suivant montre comment utiliser`Get-CFGConfigurationRecorderStatus`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie l'état des enregistreurs de configuration.

```
Get-CFGConfigurationRecorderStatus
```

Sortie :

```
LastErrorCode        :
LastErrorMessage     :
LastStartTime        : 10/11/2019 10:13:51 AM
LastStatus           : Success
LastStatusChangeTime : 12/31/2019 6:14:12 AM
LastStopTime         : 10/11/2019 10:13:46 AM
Name                 : default
Recording            : True
```

- Pour API plus de détails, consultez la section DescribeConfigurationRecorderStatusRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFGConformancePack`

L'exemple de code suivant montre comment utiliser`Get-CFGConformancePack`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie tous les packs de conformité.

```
Get-CFGConformancePack
```

Sortie :

```
ConformancePackArn             : arn:aws:config:eu-west-1:123456789012:conformance-
pack/dono/conformance-pack-p0acq8bpz
ConformancePackId              : conformance-pack-p0acabcde
ConformancePackInputParameters : {}
ConformancePackName            : dono
CreatedBy                      :
DeliveryS3Bucket               : kt-ps-examples
DeliveryS3KeyPrefix            :
LastUpdateRequestedTime        : 12/31/2019 8:45:31 AM
```

- Pour API plus de détails, consultez la section DescribeConformancePacksRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFGDeliveryChannel`

L'exemple de code suivant montre comment utiliser`Get-CFGDeliveryChannel`.

Outils pour PowerShell

Exemple 1 : Cet exemple extrait le canal de distribution de la région et affiche les détails.

```
Get-CFGDeliveryChannel -Region eu-west-1 | Select-Object Name, S3BucketName,
 S3KeyPrefix,
 @{N="DeliveryFrequency";E={$_.ConfigSnapshotDeliveryProperties.DeliveryFrequency}}
```

Sortie :

```
Name    S3BucketName              S3KeyPrefix DeliveryFrequency
----    ------------              ----------- -----------------
default config-bucket-NA my         TwentyFour_Hours
```

- Pour API plus de détails, consultez la section DescribeDeliveryChannelsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-CFGResourceTag`

L'exemple de code suivant montre comment utiliser`Get-CFGResourceTag`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie les balises associées à la ressource donnée

```
Get-CFGResourceTag -ResourceArn $rules[0].ConfigRuleArn
```

Sortie :

```
Key     Value
---     -----
Version 1.3
```

- Pour API plus de détails, consultez la section ListTagsForResourceRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-CFGConformancePack`

L'exemple de code suivant montre comment utiliser`Remove-CFGConformancePack`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le pack de conformité donné, ainsi que toutes les règles, actions correctives et résultats d'évaluation du pack.

```
Remove-CFGConformancePack -ConformancePackName dono
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-CFGConformancePack (DeleteConformancePack)" on
 target "dono".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Pour API plus de détails, consultez la section DeleteConformancePackRéférence des AWS Tools for PowerShell applets de commande.

### `Write-CFGConformancePack`

L'exemple de code suivant montre comment utiliser`Write-CFGConformancePack`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un pack de conformité en récupérant le modèle à partir du fichier yaml donné.

```
Write-CFGConformancePack -ConformancePackName dono -DeliveryS3Bucket amzn-s3-demo-
bucket -TemplateBody (Get-Content C:\windows\temp\template.yaml -Raw)
```

- Pour API plus de détails, consultez la section PutConformancePackRéférence des AWS Tools for PowerShell applets de commande.

### `Write-CFGDeliveryChannel`

L'exemple de code suivant montre comment utiliser`Write-CFGDeliveryChannel`.

Outils pour PowerShell

Exemple 1 : Cet exemple modifie les deliveryFrequency propriétés d'un canal de distribution existant.

```
Write-CFGDeliveryChannel -ConfigSnapshotDeliveryProperties_DeliveryFrequency
 TwentyFour_Hours -DeliveryChannelName default -DeliveryChannel_S3BucketName amzn-
s3-demo-bucket -DeliveryChannel_S3KeyPrefix my
```

- Pour API plus de détails, consultez la section PutDeliveryChannelRéférence des AWS Tools for PowerShell applets de commande.

## Exemples de Device Farm utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide de AWS Tools for PowerShell with Device Farm.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `New-DFUpload`

L'exemple de code suivant montre comment utiliser`New-DFUpload`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un téléchargement de AWS Device Farm pour une application Android. Vous pouvez obtenir le projet ARN à partir de la sortie de New- DFProject ou Get-DFProjectList. Utilisez la DFUpload sortie Signed URL in New- pour télécharger un fichier sur Device Farm.

```
New-DFUpload -ContentType "application/octet-stream" -ProjectArn
 "arn:aws:devicefarm:us-west-2:123456789012:project:EXAMPLEa-7ec1-4741-9c1f-
d3e04EXAMPLE" -Name "app.apk" -Type ANDROID_APP
```

- Pour API plus de détails, consultez la section CreateUploadRéférence des AWS Tools for PowerShell applets de commande.

# AWS Directory Service exemples utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS Tools for PowerShell with AWS Directory Service.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques



# Actions

### `Add-DSIpRoute`

L'exemple de code suivant montre comment utiliser`Add-DSIpRoute`.

Outils pour PowerShell

Exemple 1 : Cette commande supprime le tag de ressource attribué à l'ID de répertoire spécifié

```
Add-DSIpRoute -DirectoryId d-123456ijkl -IpRoute @{CidrIp ="203.0.113.5/32"} -
UpdateSecurityGroupForDirectoryController $true
```

- Pour API plus de détails, consultez la section AddIpRoutesRéférence des AWS Tools for PowerShell applets de commande.

### `Add-DSResourceTag`

L'exemple de code suivant montre comment utiliser`Add-DSResourceTag`.

Outils pour PowerShell

Exemple 1 : Cette commande ajoute la balise de ressource à l'ID de répertoire spécifié

```
Add-DSResourceTag -ResourceId d-123456ijkl -Tag @{Key="myTag"; Value="mytgValue"}
```

- Pour API plus de détails, consultez la section AddTagsToResourceRéférence des AWS Tools for PowerShell applets de commande.

### `Approve-DSTrust`

L'exemple de code suivant montre comment utiliser`Approve-DSTrust`.

Outils pour PowerShell

Exemple 1 : Cet exemple appelle l' VerifyTrust APIopération AWS Directory Service pour le Trustid spécifié.

```
Approve-DSTrust -TrustId t-9067157123
```

- Pour API plus de détails, consultez la section VerifyTrustRéférence des AWS Tools for PowerShell applets de commande.

### `Confirm-DSSharedDirectory`

L'exemple de code suivant montre comment utiliser`Confirm-DSSharedDirectory`.

Outils pour PowerShell

Exemple 1 : Cet exemple accepte une demande de partage de répertoire envoyée par le propriétaire du répertoire Compte AWS.

```
Confirm-DSSharedDirectory -SharedDirectoryId d-9067012345
```

Sortie :

```
CreatedDateTime     : 12/30/2019 4:20:27 AM
LastUpdatedDateTime : 12/30/2019 4:21:40 AM
OwnerAccountId      : 123456781234
OwnerDirectoryId    : d-123456ijkl
SharedAccountId     : 123456784321
SharedDirectoryId   : d-9067012345
ShareMethod         :
ShareNotes          : This is test sharing
ShareStatus         : Sharing
```

- Pour API plus de détails, consultez la section AcceptSharedDirectoryRéférence des AWS Tools for PowerShell applets de commande.

### `Connect-DSDirectory`

L'exemple de code suivant montre comment utiliser`Connect-DSDirectory`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un AD Connector pour se connecter à un annuaire local.

```
Connect-DSDirectory -Name contoso.com -ConnectSettings_CustomerUserName
 Administrator -Password $Password -ConnectSettings_CustomerDnsIp 172.31.36.96
```

```
 -ShortName CONTOSO -Size Small -ConnectSettings_VpcId vpc-123459da -
ConnectSettings_SubnetId subnet-1234ccaa, subnet-5678ffbb
```

- Pour API plus de détails, consultez la section ConnectDirectoryRéférence des AWS Tools for PowerShell applets de commande.

### `Deny-DSSharedDirectory`

L'exemple de code suivant montre comment utiliser`Deny-DSSharedDirectory`.

Outils pour PowerShell

Exemple 1 : Cet exemple rejette une demande de partage d'annuaire envoyée depuis le compte du propriétaire de l'annuaire.

```
Deny-DSSharedDirectory -SharedDirectoryId d-9067012345
```

Sortie :

```
d-9067012345
```

- Pour API plus de détails, consultez la section RejectSharedDirectoryRéférence des AWS Tools for PowerShell applets de commande.

### `Disable-DSDirectoryShare`

L'exemple de code suivant montre comment utiliser`Disable-DSDirectoryShare`.

Outils pour PowerShell

Exemple 1 : Cet exemple arrête le partage de répertoire entre le propriétaire du répertoire et le compte du consommateur.

```
Disable-DSDirectoryShare -DirectoryId d-123456ijkl -UnshareTarget_Id 123456784321 -
UnshareTarget_Type ACCOUNT
```

Sortie :

```
d-9067012345
```

- Pour API plus de détails, consultez la section UnshareDirectoryRéférence des AWS Tools for PowerShell applets de commande.

### `Disable-DSLDAPS`

L'exemple de code suivant montre comment utiliser`Disable-DSLDAPS`.

Outils pour PowerShell

Exemple 1 : Cet exemple désactive les appels LDAP sécurisés pour le répertoire spécifié.

```
Disable-DSLDAPS -DirectoryId d-123456ijkl -Type Client
```

- Pour API plus de détails, consultez la section Désactiver LDAPS dans la référence des AWS Tools for PowerShell applets de commande.

### `Disable-DSRadius`

L'exemple de code suivant montre comment utiliser`Disable-DSRadius`.

Outils pour PowerShell

Exemple 1 : Cet exemple désactive le RADIUS serveur configuré pour un AD Connector ou un annuaire Microsoft AD.

```
Disable-DSRadius -DirectoryId d-123456ijkl
```

- Pour API plus de détails, consultez la section DisableRadiusRéférence des AWS Tools for PowerShell applets de commande.

### `Disable-DSSso`

L'exemple de code suivant montre comment utiliser`Disable-DSSso`.

Outils pour PowerShell

Exemple 1 : Cet exemple désactive l'authentification unique pour un annuaire.

```
Disable-DSSso -DirectoryId d-123456ijkl
```

- Pour API plus de détails, consultez la section DisableSsoRéférence des AWS Tools for PowerShell applets de commande.

### `Enable-DSDirectoryShare`

L'exemple de code suivant montre comment utiliser`Enable-DSDirectoryShare`.

Outils pour PowerShell

Exemple 1 : Cet exemple partage un répertoire spécifique de votre AWS compte avec un autre AWS compte en utilisant la méthode Handshake.

```
Enable-DSDirectoryShare -DirectoryId d-123456ijkl -ShareTarget_Id 123456784321 -
ShareMethod HANDSHAKE -ShareTarget_Type ACCOUNT
```

Sortie :

```
d-9067012345
```

- Pour API plus de détails, consultez la section ShareDirectoryRéférence des AWS Tools for PowerShell applets de commande.

### `Enable-DSLDAPS`

L'exemple de code suivant montre comment utiliser`Enable-DSLDAPS`.

Outils pour PowerShell

Exemple 1 : Cet exemple active le commutateur pour le répertoire spécifique afin de toujours utiliser les appels LDAP sécurisés.

```
Enable-DSLDAPS -DirectoryId d-123456ijkl -Type Client
```

- Pour API plus de détails, consultez la section Activer LDAPS dans la référence des AWS Tools for PowerShell applets de commande.

### `Enable-DSRadius`

L'exemple de code suivant montre comment utiliser`Enable-DSRadius`.

Outils pour PowerShell

Exemple 1 : Cet exemple active l'authentification multifactorielle (MFA) avec la configuration de RADIUS serveur fournie pour un AD Connector ou un annuaire Microsoft AD.

```
Enable-DSRadius -DirectoryId d-123456ijkl
-RadiusSettings_AuthenticationProtocol PAP
-RadiusSettings_DisplayLabel Radius
-RadiusSettings_RadiusPort 1812
-RadiusSettings_RadiusRetry 4
-RadiusSettings_RadiusServer 10.4.185.113
-RadiusSettings_RadiusTimeout 50
-RadiusSettings_SharedSecret wJalrXUtnFEMI
```

- Pour API plus de détails, consultez la section EnableRadiusRéférence des AWS Tools for PowerShell applets de commande.

### `Enable-DSSso`

L'exemple de code suivant montre comment utiliser`Enable-DSSso`.

Outils pour PowerShell

Exemple 1 : Cet exemple active l'authentification unique pour un annuaire.

```
Enable-DSSso -DirectoryId d-123456ijkl
```

- Pour API plus de détails, consultez la section EnableSsoRéférence des AWS Tools for PowerShell applets de commande.

### `Get-DSCertificate`

L'exemple de code suivant montre comment utiliser`Get-DSCertificate`.

Outils pour PowerShell

Exemple 1 : Cet exemple affiche des informations sur le certificat enregistré pour une LDAP connexion sécurisée.

```
Get-DSCertificate -DirectoryId d-123456ijkl -CertificateId c-906731e34f
```

Sortie :

```
CertificateId      : c-906731e34f
CommonName         : contoso-EC2AMAZ-CTGG2NM-CA
ExpiryDateTime     : 4/15/2025 6:34:15 PM
RegisteredDateTime : 4/15/2020 6:38:56 PM
State              : Registered
StateReason        : Certificate registered successfully.
```

- Pour API plus de détails, consultez la section DescribeCertificateRéférence des AWS Tools for PowerShell applets de commande.

### `Get-DSCertificateList`

L'exemple de code suivant montre comment utiliser`Get-DSCertificateList`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie tous les certificats enregistrés pour une LDAP connexion sécurisée pour le répertoire spécifié.

```
Get-DSCertificateList -DirectoryId d-123456ijkl
```

Sortie :

```
CertificateId CommonName                ExpiryDateTime       State
------------- ----------                --------------       -----
c-906731e34f  contoso-EC2AMAZ-CTGG2NM-CA 4/15/2025 6:34:15 PM Registered
```

- Pour API plus de détails, consultez la section ListCertificatesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-DSConditionalForwarder`

L'exemple de code suivant montre comment utiliser`Get-DSConditionalForwarder`.

Outils pour PowerShell

Exemple 1 : Cette commande obtient tous les redirecteurs conditionnels configurés ayant un ID de répertoire donné.

```
Get-DSConditionalForwarder -DirectoryId d-123456ijkl
```

Sortie :

```
DnsIpAddrs      RemoteDomainName ReplicationScope
----------      ---------------- ----------------
{172.31.77.239} contoso.com      Domain
```

- Pour API plus de détails, consultez la section DescribeConditionalForwardersRéférence des AWS Tools for PowerShell applets de commande.

### `Get-DSDirectory`

L'exemple de code suivant montre comment utiliser`Get-DSDirectory`.

Outils pour PowerShell

Exemple 1 : Cette commande obtient des informations sur les répertoires appartenant à ce compte.

```
Get-DSDirectory | Select-Object DirectoryId, Name, DnsIpAddrs, Type
```

Sortie :

```
DirectoryId  Name           DnsIpAddrs                    Type
-----------  ----           ----------                    ----
d-123456abcd abcd.example.com {172.31.74.189, 172.31.13.145} SimpleAD
d-123456efgh wifi.example.com {172.31.16.108, 172.31.10.56}  ADConnector
d-123456ijkl lan2.example.com {172.31.10.56, 172.31.16.108}  MicrosoftAD
```

- Pour API plus de détails, consultez la section DescribeDirectoriesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-DSDirectoryLimit`

L'exemple de code suivant montre comment utiliser`Get-DSDirectoryLimit`.

Outils pour PowerShell

Exemple 1 : Cet exemple affiche les informations relatives aux limites de répertoire pour la région us-east-1.

```
Get-DSDirectoryLimit -Region us-east-1
```

Sortie :

```
CloudOnlyDirectoriesCurrentCount : 1
CloudOnlyDirectoriesLimit        : 10
CloudOnlyDirectoriesLimitReached : False
CloudOnlyMicrosoftADCurrentCount : 1
CloudOnlyMicrosoftADLimit        : 20
CloudOnlyMicrosoftADLimitReached : False
ConnectedDirectoriesCurrentCount : 1
ConnectedDirectoriesLimit        : 10
```

- Pour API plus de détails, consultez la section GetDirectoryLimitsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-DSDomainControllerList`

L'exemple de code suivant montre comment utiliser`Get-DSDomainControllerList`.

Outils pour PowerShell

Exemple 1 : Cette commande obtient la liste détaillée des contrôleurs de domaine lancés pour l'identifiant de répertoire mentionné

```
Get-DSDomainControllerList -DirectoryId d-123456ijkl
```

Sortie :

```
AvailabilityZone         : us-east-1b
DirectoryId              : d-123456ijkl
DnsIpAddr                : 172.31.16.108
DomainControllerId       : dc-1234567aa6
LaunchTime               : 4/4/2019 4:53:43 AM
```

```
Status                    : Active
StatusLastUpdatedDateTime : 4/24/2019 1:37:54 PM
StatusReason              :
SubnetId                  : subnet-1234kkaa
VpcId                     : vpc-123459d

AvailabilityZone          : us-east-1d
DirectoryId               : d-123456ijkl
DnsIpAddr                 : 172.31.10.56
DomainControllerId        : dc-1234567aa7
LaunchTime                : 4/4/2019 4:53:43 AM
Status                    : Active
StatusLastUpdatedDateTime : 4/4/2019 5:14:31 AM
StatusReason              :
SubnetId                  : subnet-5678ffbb
VpcId                     : vpc-123459d
```

- Pour API plus de détails, consultez la section DescribeDomainControllersRéférence des AWS Tools for PowerShell applets de commande.

### `Get-DSEventTopic`

L'exemple de code suivant montre comment utiliser`Get-DSEventTopic`.

Outils pour PowerShell

Exemple 1 : Cette commande affiche les informations de la SNS rubrique configurée à des fins de notification lorsque le statut du répertoire change.

```
Get-DSEventTopic -DirectoryId d-123456ijkl
```

Sortie :

```
CreatedDateTime : 12/13/2019 11:15:32 AM
DirectoryId     : d-123456ijkl
Status          : Registered
TopicArn        : arn:aws:sns:us-east-1:123456781234:snstopicname
TopicName       : snstopicname
```

- Pour API plus de détails, consultez la section DescribeEventTopicsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-DSIpRouteList`

L'exemple de code suivant montre comment utiliser`Get-DSIpRouteList`.

Outils pour PowerShell

Exemple 1 : Cette commande obtient les blocs d'adresses IP publiques configurés dans Directory IP Routing

```
Get-DSIpRouteList -DirectoryId d-123456ijkl
```

Sortie :

```
AddedDateTime       : 12/13/2019 12:27:22 PM
CidrIp              : 203.0.113.5/32
Description         : Public IP of On-Prem DNS Server
DirectoryId         : d-123456ijkl
IpRouteStatusMsg    : Added
IpRouteStatusReason :
```

- Pour API plus de détails, consultez la section ListIpRoutesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-DSLDAPSSetting`

L'exemple de code suivant montre comment utiliser`Get-DSLDAPSSetting`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit l'état de LDAP sécurité du répertoire spécifié.

```
Get-DSLDAPSSetting -DirectoryId d-123456ijkl
```

Sortie :

```
LastUpdatedDateTime  LDAPSStatus LDAPSStatusReason
-------------------  ----------- -----------------
4/15/2020 6:51:03 PM Enabled     LDAPS is enabled successfully.
```

- Pour API plus de détails, voir D escribeLDAPSSettings dans la référence des AWS Tools for PowerShell applets de commande.

### `Get-DSLogSubscriptionList`

L'exemple de code suivant montre comment utiliser`Get-DSLogSubscriptionList`.

Outils pour PowerShell

Exemple 1 : Cette commande obtient les informations d'abonnement au journal correspondant à l'identifiant de répertoire spécifié

```
Get-DSLogSubscriptionList -DirectoryId d-123456ijkl
```

Sortie :

```
DirectoryId  LogGroupName
 SubscriptionCreatedDateTime
-----------  ------------
 --------------------------
d-123456ijkl /aws/directoryservice/d-123456ijkl-lan2.example.com 12/14/2019 9:05:23
 AM
```

- Pour API plus de détails, consultez la section ListLogSubscriptionsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-DSResourceTag`

L'exemple de code suivant montre comment utiliser`Get-DSResourceTag`.

Outils pour PowerShell

Exemple 1 : Cette commande obtient tous les tags du répertoire spécifié.

```
Get-DSResourceTag -ResourceId d-123456ijkl
```

Sortie :

```
Key   Value
---   -----
myTag myTagValue
```

- Pour API plus de détails, consultez la section ListTagsForResourceRéférence des AWS Tools for PowerShell applets de commande.

### `Get-DSSchemaExtension`

L'exemple de code suivant montre comment utiliser`Get-DSSchemaExtension`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie toutes les extensions de schéma appliquées à un annuaire Microsoft AD.

```
Get-DSSchemaExtension -DirectoryId d-123456ijkl
```

Sortie :

```
Description                : ManagedADSchemaExtension
DirectoryId                : d-123456ijkl
EndDateTime                : 4/12/2020 10:30:49 AM
SchemaExtensionId          : e-9067306643
SchemaExtensionStatus      : Completed
SchemaExtensionStatusReason : Schema updates are complete.
StartDateTime              : 4/12/2020 10:28:42 AM
```

- Pour API plus de détails, consultez la section ListSchemaExtensionsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-DSSharedDirectory`

L'exemple de code suivant montre comment utiliser`Get-DSSharedDirectory`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir les répertoires partagés de votre AWS compte

```
Get-DSSharedDirectory -OwnerDirectoryId d-123456ijkl -SharedDirectoryId d-9067012345
```

Sortie :

```
CreatedDateTime     : 12/30/2019 4:34:37 AM
LastUpdatedDateTime : 12/30/2019 4:35:22 AM
OwnerAccountId      : 123456781234
OwnerDirectoryId    : d-123456ijkl
SharedAccountId     : 123456784321
```

```
SharedDirectoryId   : d-9067012345
ShareMethod         : HANDSHAKE
ShareNotes          : This is a test Sharing
ShareStatus         : Shared
```

- Pour API plus de détails, consultez la section DescribeSharedDirectoriesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-DSSnapshot`

L'exemple de code suivant montre comment utiliser`Get-DSSnapshot`.

Outils pour PowerShell

Exemple 1 : Cette commande obtient des informations sur les instantanés de répertoire spécifiés qui appartiennent à ce compte.

```
Get-DSSnapshot -DirectoryId d-123456ijkl
```

Sortie :

```
DirectoryId : d-123456ijkl
Name        :
SnapshotId  : s-9064bd1234
StartTime   : 12/13/2019 6:33:01 PM
Status      : Completed
Type        : Auto

DirectoryId : d-123456ijkl
Name        :
SnapshotId  : s-9064bb4321
StartTime   : 12/9/2019 9:48:11 PM
Status      : Completed
Type        : Auto
```

- Pour API plus de détails, consultez la section DescribeSnapshotsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-DSSnapshotLimit`

L'exemple de code suivant montre comment utiliser`Get-DSSnapshotLimit`.

Outils pour PowerShell

Exemple 1 : Cette commande obtient les limites de captures d'écran manuelles pour un répertoire spécifié.

```
Get-DSSnapshotLimit -DirectoryId d-123456ijkl
```

Sortie :

```
ManualSnapshotsCurrentCount ManualSnapshotsLimit ManualSnapshotsLimitReached
--------------------------- -------------------- --------------------------
0                           5                    False
```

- Pour API plus de détails, consultez la section GetSnapshotLimitsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-DSTrust`

L'exemple de code suivant montre comment utiliser`Get-DSTrust`.

Outils pour PowerShell

Exemple 1 : Cette commande obtient les informations des relations de confiance créées pour l'identifiant de répertoire spécifié.

```
Get-DSTrust -DirectoryId d-123456abcd
```

Sortie :

```
CreatedDateTime          : 7/5/2019 4:55:42 AM
DirectoryId              : d-123456abcd
LastUpdatedDateTime      : 7/5/2019 4:56:04 AM
RemoteDomainName         : contoso.com
SelectiveAuth            : Disabled
StateLastUpdatedDateTime : 7/5/2019 4:56:04 AM
TrustDirection           : One-Way: Incoming
TrustId                  : t-9067157123
TrustState               : Created
TrustStateReason         :
```

```
TrustType                : Forest
```

- Pour API plus de détails, consultez la section DescribeTrustsRéférence des AWS Tools for PowerShell applets de commande.

### `New-DSAlias`

L'exemple de code suivant montre comment utiliser`New-DSAlias`.

Outils pour PowerShell

Exemple 1 : Cette commande crée un alias pour un répertoire et assigne cet alias à l'identifiant de répertoire spécifié.

```
New-DSAlias -DirectoryId d-123456ijkl -Alias MyOrgName
```

Sortie :

```
Alias     DirectoryId
-----     -----------
myorgname d-123456ijkl
```

- Pour API plus de détails, consultez la section CreateAliasRéférence des AWS Tools for PowerShell applets de commande.

### `New-DSComputer`

L'exemple de code suivant montre comment utiliser`New-DSComputer`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un nouvel objet informatique Active Directory.

```
New-DSComputer -DirectoryId d-123456ijkl -ComputerName ADMemberServer -Password
 $Password
```

Sortie :

```
ComputerAttributes                ComputerId
 ComputerName
```

```
------------------                  ----------
 ------------
{WindowsSamName, DistinguishedName} S-1-5-21-1191241402-978882507-2717148213-1662
 ADMemberServer
```

- Pour API plus de détails, consultez la section CreateComputerRéférence des AWS Tools for PowerShell applets de commande.

### `New-DSConditionalForwarder`

L'exemple de code suivant montre comment utiliser`New-DSConditionalForwarder`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un redirecteur conditionnel dans l'ID de AWS répertoire spécifié.

```
New-DSConditionalForwarder -DirectoryId d-123456ijkl -DnsIpAddr
 172.31.36.96,172.31.10.56 -RemoteDomainName contoso.com
```

- Pour API plus de détails, consultez la section CreateConditionalForwarderRéférence des AWS Tools for PowerShell applets de commande.

### `New-DSDirectory`

L'exemple de code suivant montre comment utiliser`New-DSDirectory`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un nouveau répertoire Simple AD.

```
New-DSDirectory -Name corp.example.com -Password $Password -Size Small -
VpcSettings_VpcId vpc-123459d -VpcSettings_SubnetIds subnet-1234kkaa,subnet-5678ffbb
```

- Pour API plus de détails, consultez la section CreateDirectoryRéférence des AWS Tools for PowerShell applets de commande.

### `New-DSLogSubscription`

L'exemple de code suivant montre comment utiliser`New-DSLogSubscription`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un abonnement pour transférer les journaux de sécurité du contrôleur de domaine Directory Service en temps réel vers le groupe de CloudWatch journaux Amazon spécifié dans votre Compte AWS.

```
New-DSLogSubscription -DirectoryId d-123456ijkl -LogGroupName /aws/directoryservice/
d-123456ijkl-lan2.example.com
```

- Pour API plus de détails, consultez la section CreateLogSubscriptionRéférence des AWS Tools for PowerShell applets de commande.

### `New-DSMicrosoftAD`

L'exemple de code suivant montre comment utiliser`New-DSMicrosoftAD`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un nouveau répertoire Microsoft AD dans AWS Cloud.

```
New-DSMicrosoftAD -Name corp.example.com -Password $Password -edition Standard -
VpcSettings_VpcId vpc-123459d -VpcSettings_SubnetIds subnet-1234kkaa,subnet-5678ffbb
```

- Pour API plus de détails, voir CreateMicrosoftAD dans le manuel de référence des AWS Tools for PowerShell applets de commande.

### `New-DSSnapshot`

L'exemple de code suivant montre comment utiliser`New-DSSnapshot`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un instantané de répertoire

```
New-DSSnapshot -DirectoryId d-123456ijkl
```

- Pour API plus de détails, consultez la section CreateSnapshotRéférence des AWS Tools for PowerShell applets de commande.

### `New-DSTrust`

L'exemple de code suivant montre comment utiliser`New-DSTrust`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une confiance bidirectionnelle à l'échelle de Forestwide entre votre annuaire AWS Microsoft AD géré et Microsoft Active Directory existant sur site.

```
New-DSTrust -DirectoryId d-123456ijkl -RemoteDomainName contoso.com -TrustDirection
 Two-Way -TrustType Forest -TrustPassword $Password -ConditionalForwarderIpAddr
 172.31.36.96
```

Sortie :

```
t-9067157123
```

- Pour API plus de détails, consultez la section CreateTrustRéférence des AWS Tools for PowerShell applets de commande.

### `Register-DSCertificate`

L'exemple de code suivant montre comment utiliser`Register-DSCertificate`.

Outils pour PowerShell

Exemple 1 : Cet exemple enregistre un certificat pour une LDAP connexion sécurisée.

```
$Certificate = Get-Content contoso.cer -Raw
Register-DSCertificate -DirectoryId d-123456ijkl -CertificateData $Certificate
```

Sortie :

```
c-906731e350
```

- Pour API plus de détails, consultez la section RegisterCertificateRéférence des AWS Tools for PowerShell applets de commande.

### `Register-DSEventTopic`

L'exemple de code suivant montre comment utiliser`Register-DSEventTopic`.

Outils pour PowerShell

Exemple 1 : Cet exemple associe un annuaire en tant qu'éditeur à une SNS rubrique.

```
Register-DSEventTopic -DirectoryId d-123456ijkl -TopicName snstopicname
```

- Pour API plus de détails, consultez la section RegisterEventTopicRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-DSConditionalForwarder`

L'exemple de code suivant montre comment utiliser`Remove-DSConditionalForwarder`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le redirecteur conditionnel qui a été configuré pour votre AWS répertoire.

```
Remove-DSConditionalForwarder -DirectoryId d-123456ijkl -RemoteDomainName
 contoso.com
```

- Pour API plus de détails, consultez la section DeleteConditionalForwarderRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-DSDirectory`

L'exemple de code suivant montre comment utiliser`Remove-DSDirectory`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime un AWS annuaire du service d'annuaire (connecteur AD/ Microsoft AD/AD simple)

```
Remove-DSDirectory -DirectoryId d-123456ijkl
```

- Pour API plus de détails, consultez la section DeleteDirectoryRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-DSIpRoute`

L'exemple de code suivant montre comment utiliser`Remove-DSIpRoute`.

Outils pour PowerShell

Exemple 1 : Cette commande supprime l'adresse IP spécifiée des routes IP configurées de Directory-ID.

```
Remove-DSIpRoute -DirectoryId d-123456ijkl -CidrIp 203.0.113.5/32
```

- Pour API plus de détails, consultez la section RemoveIpRoutesRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-DSLogSubscription`

L'exemple de code suivant montre comment utiliser`Remove-DSLogSubscription`.

Outils pour PowerShell

Exemple 1 : Cette commande supprime l'abonnement au journal correspondant à l'ID de répertoire spécifié

```
Remove-DSLogSubscription -DirectoryId d-123456ijkl
```

- Pour API plus de détails, consultez la section DeleteLogSubscriptionRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-DSResourceTag`

L'exemple de code suivant montre comment utiliser`Remove-DSResourceTag`.

Outils pour PowerShell

Exemple 1 : Cette commande supprime le tag de ressource attribué à l'ID de répertoire spécifié

```
Remove-DSResourceTag -ResourceId d-123456ijkl -TagKey myTag
```

- Pour API plus de détails, consultez la section RemoveTagsFromResourceRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-DSSnapshot`

L'exemple de code suivant montre comment utiliser`Remove-DSSnapshot`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le cliché créé manuellement.

```
Remove-DSSnapshot -SnapshotId s-9068b488kc
```

- Pour API plus de détails, consultez la section DeleteSnapshotRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-DSTrust`

L'exemple de code suivant montre comment utiliser`Remove-DSTrust`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la relation de confiance existante entre votre annuaire AD AWS géré et un domaine externe.

```
Get-DSTrust -DirectoryId d-123456ijkl -Select Trusts.TrustId | Remove-DSTrust
```

Sortie :

```
t-9067157123
```

- Pour API plus de détails, consultez la section DeleteTrustRéférence des AWS Tools for PowerShell applets de commande.

### `Reset-DSUserPassword`

L'exemple de code suivant montre comment utiliser`Reset-DSUserPassword`.

Outils pour PowerShell

Exemple 1 : Cet exemple réinitialise le mot de passe de l'utilisateur Active Directory nommé ADUser dans AWS Managed Microsoft AD ou Simple AD Directory

```
Reset-DSUserPassword -UserName ADuser -DirectoryId d-123456ijkl -NewPassword
 $Password
```

- Pour API plus de détails, consultez la section ResetUserPasswordRéférence des AWS Tools for PowerShell applets de commande.

### `Restore-DSFromSnapshot`

L'exemple de code suivant montre comment utiliser`Restore-DSFromSnapshot`.

Outils pour PowerShell

Exemple 1 : Cet exemple restaure un répertoire à l'aide d'un instantané de répertoire existant.

```
Restore-DSFromSnapshot -SnapshotId s-9068b488kc
```

- Pour API plus de détails, consultez la section RestoreFromSnapshotRéférence des AWS Tools for PowerShell applets de commande.

### `Set-DSDomainControllerCount`

L'exemple de code suivant montre comment utiliser`Set-DSDomainControllerCount`.

Outils pour PowerShell

Exemple 1 : Cet exemple définit le nombre de contrôleurs de domaine à 3 pour l'identifiant de répertoire spécifié.

```
Set-DSDomainControllerCount -DirectoryId d-123456ijkl -DesiredNumber 3
```

- Pour API plus de détails, consultez la section UpdateNumberOfDomainControllersRéférence des AWS Tools for PowerShell applets de commande.

### `Start-DSSchemaExtension`

L'exemple de code suivant montre comment utiliser`Start-DSSchemaExtension`.

Outils pour PowerShell

Exemple 1 : Cet exemple applique une extension de schéma à un annuaire Microsoft AD.

```
$ldif = Get-Content D:\Users\Username\Downloads\ExtendedSchema.ldf -Raw
Start-DSSchemaExtension -DirectoryId d-123456ijkl -
CreateSnapshotBeforeSchemaExtension $true -Description ManagedADSchemaExtension -
LdifContent $ldif
```

Sortie :

```
e-9067306643
```

- Pour API plus de détails, consultez la section StartSchemaExtensionRéférence des AWS Tools for PowerShell applets de commande.

### `Stop-DSSchemaExtension`

L'exemple de code suivant montre comment utiliser`Stop-DSSchemaExtension`.

Outils pour PowerShell

Exemple 1 : Cet exemple annule une extension de schéma en cours vers un annuaire Microsoft AD.

```
Stop-DSSchemaExtension -DirectoryId d-123456ijkl -SchemaExtensionId e-9067306643
```

- Pour API plus de détails, consultez la section CancelSchemaExtensionRéférence des AWS Tools for PowerShell applets de commande.

### `Unregister-DSCertificate`

L'exemple de code suivant montre comment utiliser`Unregister-DSCertificate`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime du système le certificat enregistré pour une LDAP connexion sécurisée.

```
Unregister-DSCertificate -DirectoryId d-123456ijkl -CertificateId c-906731e34f
```

- Pour API plus de détails, consultez la section DeregisterCertificateRéférence des AWS Tools for PowerShell applets de commande.

### `Unregister-DSEventTopic`

L'exemple de code suivant montre comment utiliser`Unregister-DSEventTopic`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le répertoire spécifié en tant qu'éditeur de la rubrique spécifiéeSNS.

```
Unregister-DSEventTopic -DirectoryId d-123456ijkl -TopicName snstopicname
```

- Pour API plus de détails, consultez la section DeregisterEventTopicRéférence des AWS Tools for PowerShell applets de commande.

### `Update-DSConditionalForwarder`

L'exemple de code suivant montre comment utiliser`Update-DSConditionalForwarder`.

Outils pour PowerShell

Exemple 1 : Cet exemple met à jour un redirecteur conditionnel qui a été configuré pour votre AWS annuaire.

```
Update-DSConditionalForwarder -DirectoryId d-123456ijkl -DnsIpAddr
 172.31.36.96,172.31.16.108 -RemoteDomainName contoso.com
```

- Pour API plus de détails, consultez la section UpdateConditionalForwarderRéférence des AWS Tools for PowerShell applets de commande.

### `Update-DSRadius`

L'exemple de code suivant montre comment utiliser`Update-DSRadius`.

Outils pour PowerShell

Exemple 1 : Cet exemple met à jour les informations RADIUS du serveur pour un AD Connector ou un annuaire Microsoft AD.

```
Update-DSRadius -DirectoryId d-123456ijkl -RadiusSettings_RadiusRetry 3
```

- Pour API plus de détails, consultez la section UpdateRadiusRéférence des AWS Tools for PowerShell applets de commande.

### `Update-DSTrust`

L'exemple de code suivant montre comment utiliser`Update-DSTrust`.

Outils pour PowerShell

Exemple 1 : Cet exemple met à jour le SelectiveAuth paramètre de l'identifiant de confiance spécifié de Disabled à Enabled.

```
Update-DSTrust -TrustId t-9067157123 -SelectiveAuth Enabled
```

Sortie :

```
RequestId                            TrustId
---------                            -------
138864a7-c9a8-4ad1-a828-eae479e85b45 t-9067157123
```

- Pour API plus de détails, consultez la section UpdateTrustRéférence des AWS Tools for PowerShell applets de commande.

## AWS DMS exemples utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS Tools for PowerShell with AWS DMS.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `New-DMSReplicationTask`

L'exemple de code suivant montre comment utiliser`New-DMSReplicationTask`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une nouvelle tâche de réplication du Service AWS de migration de base de données qui utilise CdcStartTime au lieu de CdcStartPosition. Le MigrationType est défini sur full-load-and-cdc « », ce qui signifie que la table cible doit être vide. La nouvelle tâche est étiquetée avec une balise dont la clé est Stage et la valeur clé est Test. Pour plus d'informations sur les valeurs utilisées par cette applet de commande, consultez la section Création d'une tâche (https://docs.aws.amazon.com/dms/latest/userguide/ CHAP _Tasks.creating.html) dans le Guide de l'utilisateur du Service de Migration de Base de Données. AWS

```
New-DMSReplicationTask -ReplicationInstanceArn "arn:aws:dms:us-
east-1:123456789012:rep:EXAMPLE66XFJUWATDJGBEXAMPLE"`
  -CdcStartTime "2019-08-08T12:12:12"`
  -CdcStopPosition "server_time:2019-08-09T12:12:12"`
  -MigrationType "full-load-and-cdc"`
  -ReplicationTaskIdentifier "task1"`
  -ReplicationTaskSetting ""`
  -SourceEndpointArn "arn:aws:dms:us-
east-1:123456789012:endpoint:EXAMPLEW5UANC7Y3P4EEXAMPLE"`
  -TableMapping "file:////home/testuser/table-mappings.json"`
  -Tag @{"Key"="Stage";"Value"="Test"}`
  -TargetEndpointArn "arn:aws:dms:us-
east-1:123456789012:endpoint:EXAMPLEJZASXWHTWCLNEXAMPLE"
```

- Pour API plus de détails, consultez la section CreateReplicationTaskRéférence des AWS Tools for PowerShell applets de commande.

# Exemples DynamoDB utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS Tools for PowerShell aide de DynamoDB.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques



## Actions

### `Add-DDBIndexSchema`

L'exemple de code suivant montre comment utiliser`Add-DDBIndexSchema`.

Outils pour PowerShell

Exemple 1 : crée un TableSchema objet vide et y ajoute une nouvelle définition d'index secondaire local avant d'écrire l' TableSchema objet dans le pipeline.

```
$schema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
 "LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
$schema = New-DDBTableSchema
```

Sortie :

```
AttributeSchema                             KeySchema
    LocalSecondaryIndexSchema
---------------                             ---------
    -------------------------
{LastPostDateTime}                          {}
    {LastPostIndex}
```

Exemple 2 : ajoute une nouvelle définition d'index secondaire local à l' TableSchema objet fourni avant de réécrire l' TableSchema objet dans le pipeline. L' TableSchema objet peut également être fourni à l'aide du paramètre -Schema.

```
New-DDBTableSchema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
 "LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
```

Sortie :

```
AttributeSchema                             KeySchema
    LocalSecondaryIndexSchema
---------------                             ---------
    -------------------------
{LastPostDateTime}                          {}
    {LastPostIndex}
```

- Pour API plus de détails, consultez la section Référence de l'AWS Tools for PowerShell applet de commande Add- DDBIndexSchema in.

### `Add-DDBKeySchema`

L'exemple de code suivant montre comment utiliserAdd-DDBKeySchema.

Outils pour PowerShell

Exemple 1 : Crée un TableSchema objet vide et y ajoute des entrées de définition de clé et d'attribut à l'aide des données clés spécifiées avant d'écrire l' TableSchema objet dans le pipeline. Le type de clé est déclaré « HASH » par défaut ; utilisez le KeyType paramètre - avec une valeur de « RANGE » pour déclarer une clé de plage.

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
```

Sortie :

```
AttributeSchema                             KeySchema
    LocalSecondaryIndexSchema
---------------                             ---------
    -------------------------
{ForumName}                                 {ForumName}
    {}
```

Exemple 2 : ajoute de nouvelles entrées de définition de clé et d'attribut à l' TableSchema objet fourni avant d'écrire l' TableSchema objet dans le pipeline. Le type de clé est déclaré « HASH » par défaut ; utilisez le KeyType paramètre - avec une valeur de « RANGE » pour déclarer une clé de plage. L' TableSchema objet peut également être fourni à l'aide du paramètre -Schema.

```
New-DDBTableSchema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
```

Sortie :

```
AttributeSchema                             KeySchema
    LocalSecondaryIndexSchema
---------------                             ---------
    -------------------------
{ForumName}                                 {ForumName}
    {}
```

- Pour API plus de détails, consultez la section Référence de l'AWS Tools for PowerShell applet de commande Add- DDBKeySchema in.

### `ConvertFrom-DDBItem`

L'exemple de code suivant montre comment utiliser`ConvertFrom-DDBItem`.

Outils pour PowerShell

Exemple 1 : ConvertFrom - DDBItem est utilisé pour convertir le résultat de Get- d'une table DDBItem de hachage de AttributeValues DynamoDB en une table de hachage de types courants tels que string et double.

```
@{
    SongTitle = 'Somewhere Down The Road'
    Artist    = 'No One You Know'
} | ConvertTo-DDBItem

Get-DDBItem -TableName 'Music' -Key $key | ConvertFrom-DDBItem
```

Sortie :

```
Name                           Value
----                           -----
Genre                          Country
Artist                         No One You Know
Price                          1.94
CriticRating                   9
SongTitle                      Somewhere Down The Road
```

```
AlbumTitle                     Somewhat Famous
```

- Pour API plus de détails, reportez-vous à la section ConvertFrom- DDBItem dans la référence des AWS Tools for PowerShell applets de commande.

### `ConvertTo-DDBItem`

L'exemple de code suivant montre comment utiliser`ConvertTo-DDBItem`.

Outils pour PowerShell

Exemple 1 : exemple de conversion d'une table de hachage en dictionnaire de valeurs d'attributs DynamoDB.

```
@{
    SongTitle = 'Somewhere Down The Road'
    Artist    = 'No One You Know'
} | ConvertTo-DDBItem

Key       Value
---       -----
SongTitle Amazon.DynamoDBv2.Model.AttributeValue
Artist    Amazon.DynamoDBv2.Model.AttributeValue
```

Exemple 2 : exemple de conversion d'une table de hachage en dictionnaire de valeurs d'attributs DynamoDB.

```
@{
    MyMap        = @{
        MyString = 'my string'
    }
    MyStringSet  = [System.Collections.Generic.HashSet[String]]@('my', 'string')
    MyNumericSet = [System.Collections.Generic.HashSet[Int]]@(1, 2, 3)
    MyBinarySet  = [System.Collections.Generic.HashSet[System.IO.MemoryStream]]@(
        ([IO.MemoryStream]::new([Text.Encoding]::UTF8.GetBytes('my'))),
        ([IO.MemoryStream]::new([Text.Encoding]::UTF8.GetBytes('string')))
    )
    MyList1      = @('my', 'string')
    MyList2      = [System.Collections.Generic.List[Int]]@(1, 2)
    MyList3      = [System.Collections.ArrayList]@('one', 2, $true)
} | ConvertTo-DDBItem
```

Sortie :

```
Key          Value
---          -----
MyStringSet  Amazon.DynamoDBv2.Model.AttributeValue
MyList1      Amazon.DynamoDBv2.Model.AttributeValue
MyNumericSet Amazon.DynamoDBv2.Model.AttributeValue
MyList2      Amazon.DynamoDBv2.Model.AttributeValue
MyBinarySet  Amazon.DynamoDBv2.Model.AttributeValue
MyMap        Amazon.DynamoDBv2.Model.AttributeValue
MyList3      Amazon.DynamoDBv2.Model.AttributeValue
```

- Pour API plus de détails, reportez-vous à la section ConvertTo- DDBItem dans la référence des AWS Tools for PowerShell applets de commande.

### `Get-DDBBatchItem`

L'exemple de code suivant montre comment utiliser`Get-DDBBatchItem`.

Outils pour PowerShell

Exemple 1 : extrait l'élément SongTitle « Somewhere Down The Road » des tables DynamoDB « Music » et « Songs ».

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem

$keysAndAttributes = New-Object Amazon.DynamoDBv2.Model.KeysAndAttributes
$list = New-Object
 'System.Collections.Generic.List[System.Collections.Generic.Dictionary[String,
 Amazon.DynamoDBv2.Model.AttributeValue]]'
$list.Add($key)
$keysAndAttributes.Keys = $list

$requestItem = @{
    'Music' = [Amazon.DynamoDBv2.Model.KeysAndAttributes]$keysAndAttributes
    'Songs' = [Amazon.DynamoDBv2.Model.KeysAndAttributes]$keysAndAttributes
}

$batchItems = Get-DDBBatchItem -RequestItem $requestItem
```

```
$batchItems.GetEnumerator() | ForEach-Object {$PSItem.Value} | ConvertFrom-DDBItem
```

Sortie :

```
Name                           Value
----                           -----
Artist                         No One You Know
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
CriticRating                   10
Genre                          Country
Price                          1.94
Artist                         No One You Know
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
CriticRating                   10
Genre                          Country
Price                          1.94
```

- Pour API plus de détails, consultez la section BatchGetItemRéférence des AWS Tools for PowerShell applets de commande.

### `Get-DDBItem`

L'exemple de code suivant montre comment utiliser`Get-DDBItem`.

Outils pour PowerShell

Exemple 1 : renvoie l'élément DynamoDB avec la clé de partition et la SongTitle clé de tri Artist.

```
$key = @{
  SongTitle = 'Somewhere Down The Road'
  Artist = 'No One You Know'
} | ConvertTo-DDBItem

Get-DDBItem -TableName 'Music' -Key $key | ConvertFrom-DDBItem
```

Sortie :

```
Name                           Value
----                           -----
Genre                          Country
```

```
SongTitle                      Somewhere Down The Road
Price                          1.94
Artist                         No One You Know
CriticRating                   9
AlbumTitle                     Somewhat Famous
```

- Pour API plus de détails, consultez la section GetItemRéférence des AWS Tools for PowerShell applets de commande.

### `Get-DDBTable`

L'exemple de code suivant montre comment utiliser`Get-DDBTable`.

Outils pour PowerShell

Exemple 1 : renvoie les détails de la table spécifiée.

```
Get-DDBTable -TableName "myTable"
```

- Pour API plus de détails, consultez la section DescribeTableRéférence des AWS Tools for PowerShell applets de commande.

### `Get-DDBTableList`

L'exemple de code suivant montre comment utiliser`Get-DDBTableList`.

Outils pour PowerShell

Exemple 1 : renvoie les détails de toutes les tables, en itérant automatiquement jusqu'à ce que le service indique qu'aucune autre table n'existe.

```
Get-DDBTableList
```

Exemple 2 : itère manuellement pour obtenir les détails de toutes les tables, renvoyant jusqu'à 10 tables par appel jusqu'à ce que le service indique qu'aucune autre table n'existe.

```
$nextToken = $null
do {
  Get-DDBTableList -ExclusiveStartTableName $nextToken -Limit 10
  $nextToken = $AWSHistory.LastServiceResponse.LastEvaluatedTableName
```

```
} while ($nextToken -ne $null)
```

- Pour API plus de détails, consultez la section ListTablesRéférence des AWS Tools for PowerShell applets de commande.

### `Invoke-DDBQuery`

L'exemple de code suivant montre comment utiliser`Invoke-DDBQuery`.

Outils pour PowerShell

Exemple 1 : invoque une requête qui renvoie des éléments DynamoDB avec les valeurs spécifiées et Artist. SongTitle

```
$invokeDDBQuery = @{
    TableName = 'Music'
    KeyConditionExpression = ' SongTitle = :SongTitle and Artist = :Artist'
    ExpressionAttributeValues = @{
        ':SongTitle' = 'Somewhere Down The Road'
        ':Artist' = 'No One You Know'
    } | ConvertTo-DDBItem
}
Invoke-DDBQuery @invokeDDBQuery | ConvertFrom-DDBItem
```

Sortie :

```
Name                           Value
----                           -----
Genre                          Country
Artist                         No One You Know
Price                          1.94
CriticRating                   9
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
```

- Pour API plus de détails, voir Requête dans la référence des AWS Tools for PowerShell applets de commande.

### `Invoke-DDBScan`

L'exemple de code suivant montre comment utiliser`Invoke-DDBScan`.

Outils pour PowerShell

Exemple 1 : renvoie tous les éléments de la table Musique.

```
Invoke-DDBScan -TableName 'Music' | ConvertFrom-DDBItem
```

Sortie :

```
Name                           Value
----                           -----
Genre                          Country
Artist                         No One You Know
Price                          1.94
CriticRating                   9
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
Genre                          Country
Artist                         No One You Know
Price                          1.98
CriticRating                   8.4
SongTitle                      My Dog Spot
AlbumTitle                     Hey Now
```

Exemple 2 : renvoie les éléments du tableau Musique dont le score est CriticRating supérieur ou égal à neuf.

```
$scanFilter = @{
        CriticRating = [Amazon.DynamoDBv2.Model.Condition]@{
            AttributeValueList = @(@{N = '9'})
            ComparisonOperator = 'GE'
        }
    }
    Invoke-DDBScan -TableName 'Music' -ScanFilter $scanFilter | ConvertFrom-DDBItem
```

Sortie :

```
Name                           Value
----                           -----
Genre                          Country
Artist                         No One You Know
Price                          1.94
CriticRating                   9
```

```
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
```

- Pour API plus de détails, consultez la section Scan in AWS Tools for PowerShell Cmdlet Reference.

### `New-DDBTable`

L'exemple de code suivant montre comment utiliser`New-DDBTable`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une table nommée Thread dont la clé primaire est composée de « ForumName » (hachage de type de clé) et de « Subject » (plage de types de clés). Le schéma utilisé pour construire la table peut être redirigé vers chaque applet de commande comme indiqué ou spécifié à l'aide du paramètre -Schema.

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
$schema | Add-DDBKeySchema -KeyName "Subject" -KeyType RANGE -KeyDataType "S"
$schema | New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

Sortie :

```
AttributeDefinitions   : {ForumName, Subject}
TableName              : Thread
KeySchema              : {ForumName, Subject}
TableStatus            : CREATING
CreationDateTime       : 10/28/2013 4:39:49 PM
ProvisionedThroughput  : Amazon.DynamoDBv2.Model.ProvisionedThroughputDescription
TableSizeBytes         : 0
ItemCount              : 0
LocalSecondaryIndexes  : {}
```

Exemple 2 : Cet exemple crée une table nommée Thread dont la clé primaire est composée de « ForumName » (hachage de type de clé) et de « Subject » (plage de types de clés). Un index secondaire local est également défini. La clé de l'index secondaire local sera définie automatiquement à partir de la clé de hachage principale de la table (ForumName). Le schéma utilisé pour construire la table peut être redirigé vers chaque applet de commande comme indiqué ou spécifié à l'aide du paramètre -Schema.

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
$schema | Add-DDBKeySchema -KeyName "Subject" -KeyDataType "S"
$schema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
 "LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
$schema | New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

Sortie :

```
AttributeDefinitions   : {ForumName, LastPostDateTime, Subject}
TableName              : Thread
KeySchema              : {ForumName, Subject}
TableStatus            : CREATING
CreationDateTime       : 10/28/2013 4:39:49 PM
ProvisionedThroughput  : Amazon.DynamoDBv2.Model.ProvisionedThroughputDescription
TableSizeBytes         : 0
ItemCount              : 0
LocalSecondaryIndexes  : {LastPostIndex}
```

Exemple 3 : Cet exemple montre comment utiliser un pipeline unique pour créer une table nommée Thread qui possède une clé primaire composée de « ForumName » (hachage de type de clé) et de « Subject » (plage de types de clés) et un index secondaire local. Les options Add-DDBKeySchema et Add- DDBIndexSchema créent un nouvel TableSchema objet pour vous si aucun objet n'est fourni par le pipeline ou le paramètre -Schema.

```
New-DDBTableSchema |
  Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S" |
  Add-DDBKeySchema -KeyName "Subject" -KeyDataType "S" |
  Add-DDBIndexSchema -IndexName "LastPostIndex" `
                     -RangeKeyName "LastPostDateTime" `
                     -RangeKeyDataType "S" `
                     -ProjectionType "keys_only" |
  New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

Sortie :

```
AttributeDefinitions   : {ForumName, LastPostDateTime, Subject}
TableName              : Thread
KeySchema              : {ForumName, Subject}
TableStatus            : CREATING
CreationDateTime       : 10/28/2013 4:39:49 PM
```

```
ProvisionedThroughput  : Amazon.DynamoDBv2.Model.ProvisionedThroughputDescription
TableSizeBytes         : 0
ItemCount              : 0
LocalSecondaryIndexes  : {LastPostIndex}
```

- Pour API plus de détails, consultez la section CreateTableRéférence des AWS Tools for PowerShell applets de commande.

### `New-DDBTableSchema`

L'exemple de code suivant montre comment utiliser`New-DDBTableSchema`.

Outils pour PowerShell

Exemple 1 : crée un TableSchema objet vide prêt à accepter les définitions de clé et d'index à utiliser lors de la création d'une nouvelle table Amazon DynamoDB. L'objet renvoyé peut être redirigé vers les DDBTable applets de commande Add-DDBKeySchema, Add- DDBIndexSchema et New- ou leur être transmis à l'aide du paramètre -Schema de chaque applet de commande.

```
New-DDBTableSchema
```

Sortie :

```
AttributeSchema                             KeySchema
    LocalSecondaryIndexSchema
---------------                             ---------
    ------------------------
{}                                          {}
    {}
```

- Pour API plus de détails, consultez la section New- DDBTableSchema in AWS Tools for PowerShell Cmdlet Reference.

### `Remove-DDBItem`

L'exemple de code suivant montre comment utiliser`Remove-DDBItem`.

Outils pour PowerShell

Exemple 1 : Supprime l'élément DynamoDB correspondant à la clé fournie.

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem
Remove-DDBItem -TableName 'Music' -Key $key -Confirm:$false
```

- Pour API plus de détails, consultez la section DeleteItemRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-DDBTable`

L'exemple de code suivant montre comment utiliser`Remove-DDBTable`.

Outils pour PowerShell

Exemple 1 : Supprime la table spécifiée. Vous êtes invité à confirmer avant que l'opération ne se poursuive.

```
Remove-DDBTable -TableName "myTable"
```

Exemple 2 : Supprime la table spécifiée. Aucune confirmation ne vous est demandée avant le début de l'opération.

```
Remove-DDBTable -TableName "myTable" -Force
```

- Pour API plus de détails, consultez la section DeleteTableRéférence des AWS Tools for PowerShell applets de commande.

### `Set-DDBBatchItem`

L'exemple de code suivant montre comment utiliser`Set-DDBBatchItem`.

Outils pour PowerShell

Exemple 1 : crée un nouvel élément ou remplace un élément existant par un nouvel élément dans les tables DynamoDB Music et Songs.

```
$item = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
```

```
        AlbumTitle = 'Somewhat Famous'
        Price = 1.94
        Genre = 'Country'
        CriticRating = 10.0
} | ConvertTo-DDBItem

$writeRequest = New-Object Amazon.DynamoDBv2.Model.WriteRequest
$writeRequest.PutRequest = [Amazon.DynamoDBv2.Model.PutRequest]$item
```

Sortie :

```
$requestItem = @{
    'Music' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
    'Songs' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
}

Set-DDBBatchItem -RequestItem $requestItem
```

- Pour API plus de détails, consultez la section BatchWriteItemRéférence des AWS Tools for PowerShell applets de commande.

### `Set-DDBItem`

L'exemple de code suivant montre comment utiliser`Set-DDBItem`.

Outils pour PowerShell

Exemple 1 : crée un nouvel article ou remplace un article existant par un nouvel article.

```
$item = @{
  SongTitle = 'Somewhere Down The Road'
  Artist = 'No One You Know'
        AlbumTitle = 'Somewhat Famous'
        Price = 1.94
        Genre = 'Country'
        CriticRating = 9.0
} | ConvertTo-DDBItem
Set-DDBItem -TableName 'Music' -Item $item
```

- Pour API plus de détails, consultez la section PutItemRéférence des AWS Tools for PowerShell applets de commande.

### `Update-DDBItem`

L'exemple de code suivant montre comment utiliser`Update-DDBItem`.

Outils pour PowerShell

Exemple 1 : définit l'attribut genre sur « Rap » sur l'élément DynamoDB avec la clé de partition et la SongTitle clé de tri Artist.

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem

$updateDdbItem = @{
    TableName = 'Music'
    Key = $key
    UpdateExpression = 'set Genre = :val1'
    ExpressionAttributeValue = (@{
        ':val1' = ([Amazon.DynamoDBv2.Model.AttributeValue]'Rap')
    })
}
Update-DDBItem @updateDdbItem
```

Sortie :

```
Name                           Value
----                           -----
Genre                          Rap
```

- Pour API plus de détails, consultez la section UpdateItemRéférence des AWS Tools for PowerShell applets de commande.

### `Update-DDBTable`

L'exemple de code suivant montre comment utiliser`Update-DDBTable`.

Outils pour PowerShell

Exemple 1 : met à jour le débit provisionné pour la table donnée.

```
Update-DDBTable -TableName "myTable" -ReadCapacity 10 -WriteCapacity 5
```

- Pour API plus de détails, consultez la section UpdateTableRéférence des AWS Tools for PowerShell applets de commande.

# EC2Exemples Amazon utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS Tools for PowerShell aide d'AmazonEC2.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `Add-EC2CapacityReservation`

L'exemple de code suivant montre comment utiliser`Add-EC2CapacityReservation`.

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 API plus de détails, consultez la section CreateCapacityReservationRéférence des AWS Tools for PowerShell applets de commande.

### `Add-EC2InternetGateway`

L'exemple de code suivant montre comment utiliser`Add-EC2InternetGateway`.

Outils pour PowerShell

Exemple 1 : Cet exemple attache la passerelle Internet spécifiée à la passerelle spécifiéeVPC.

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

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

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

- Pour API plus de détails, consultez la section AttachInternetGatewayRéférence des AWS Tools for PowerShell applets de commande.

### `Add-EC2NetworkInterface`

L'exemple de code suivant montre comment utiliser`Add-EC2NetworkInterface`.

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 API plus de détails, consultez la section AttachNetworkInterfaceRéférence des AWS Tools for PowerShell applets de commande.

### `Add-EC2Volume`

L'exemple de code suivant montre comment utiliser`Add-EC2Volume`.

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 API plus de détails, consultez la section AttachVolumeRéférence des AWS Tools for PowerShell applets de commande.

### `Add-EC2VpnGateway`

L'exemple de code suivant montre comment utiliser`Add-EC2VpnGateway`.

Outils pour PowerShell

Exemple 1 : Cet exemple attache la passerelle privée virtuelle spécifiée à la passerelle spécifiéeVPC.

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

Sortie :

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

- Pour API plus de détails, consultez la section AttachVpnGatewayRéférence des AWS Tools for PowerShell applets de commande.

### `Approve-EC2VpcPeeringConnection`

L'exemple de code suivant montre comment utiliser`Approve-EC2VpcPeeringConnection`.

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 API plus de détails, consultez la section AcceptVpcPeeringConnectionRéférence des AWS Tools for PowerShell applets de commande.

### `Confirm-EC2ProductInstance`

L'exemple de code suivant montre comment utiliser`Confirm-EC2ProductInstance`.

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 API plus de détails, consultez la section ConfirmProductInstanceRéférence des AWS Tools for PowerShell applets de commande.

### `Copy-EC2Image`

L'exemple de code suivant montre comment utiliser`Copy-EC2Image`.

Outils pour PowerShell

Exemple 1 : Cet exemple copie les données spécifiées AMI 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 API plus de détails, consultez la section CopyImageRéférence des AWS Tools for PowerShell applets de commande.

### `Copy-EC2Snapshot`

L'exemple de code suivant montre comment utiliser`Copy-EC2Snapshot`.

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 API plus de détails, consultez la section CopySnapshotRéférence des AWS Tools for PowerShell applets de commande.

### `Deny-EC2VpcPeeringConnection`

L'exemple de code suivant montre comment utiliser`Deny-EC2VpcPeeringConnection`.

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 API plus de détails, consultez la section RejectVpcPeeringConnectionRéférence des AWS Tools for PowerShell applets de commande.

### `Disable-EC2VgwRoutePropagation`

L'exemple de code suivant montre comment utiliser`Disable-EC2VgwRoutePropagation`.

Outils pour PowerShell

Exemple 1 : Cet exemple désactive la propagation automatique VGW des itinéraires vers la table de routage spécifiée.

```
Disable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- Pour API plus de détails, consultez la section DisableVgwRoutePropagationRéférence des AWS Tools for PowerShell applets de commande.

### `Disable-EC2VpcClassicLink`

L'exemple de code suivant montre comment utiliser`Disable-EC2VpcClassicLink`.

Outils pour PowerShell

Exemple 1 : Cet exemple désactive le EC2VpcClassicLink vpc-01e23c4a5d6db78e9. Elle renvoie Vrai ou Faux

```
Disable-EC2VpcClassicLink -VpcId vpc-01e23c4a5d6db78e9
```

- Pour API plus de détails, consultez la section DisableVpcClassicLinkRéférence des AWS Tools for PowerShell applets de commande.

### `Disable-EC2VpcClassicLinkDnsSupport`

L'exemple de code suivant montre comment utiliser`Disable-EC2VpcClassicLinkDnsSupport`.

Outils pour PowerShell

Exemple 1 : cet exemple désactive le ClassicLink DNS support pour le vpc-0b12d3456a7e8910d

```
Disable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d
```

- Pour API plus de détails, consultez la section DisableVpcClassicLinkDnsSupportRéférence des AWS Tools for PowerShell applets de commande.

### `Dismount-EC2InternetGateway`

L'exemple de code suivant montre comment utiliser`Dismount-EC2InternetGateway`.

Outils pour PowerShell

Exemple 1 : Cet exemple détache la passerelle Internet spécifiée de la passerelle spécifiéeVPC.

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

- Pour API plus de détails, consultez la section DetachInternetGatewayRéférence des AWS Tools for PowerShell applets de commande.

### `Dismount-EC2NetworkInterface`

L'exemple de code suivant montre comment utiliser`Dismount-EC2NetworkInterface`.

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 API plus de détails, consultez la section DetachNetworkInterfaceRéférence des AWS Tools for PowerShell applets de commande.

### `Dismount-EC2Volume`

L'exemple de code suivant montre comment utiliser`Dismount-EC2Volume`.

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 API plus de détails, consultez la section DetachVolumeRéférence des AWS Tools for PowerShell applets de commande.

### `Dismount-EC2VpnGateway`

L'exemple de code suivant montre comment utiliser`Dismount-EC2VpnGateway`.

Outils pour PowerShell

Exemple 1 : Cet exemple détache la passerelle privée virtuelle spécifiée de la passerelle spécifiéeVPC.

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

- Pour API plus de détails, consultez la section DetachVpnGatewayRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-EC2CapacityReservation`

L'exemple de code suivant montre comment utiliser`Edit-EC2CapacityReservation`.

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 API plus de détails, consultez la section ModifyCapacityReservationRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-EC2Host`

L'exemple de code suivant montre comment utiliser`Edit-EC2Host`.

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 API plus de détails, consultez la section ModifyHostsRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-EC2IdFormat`

L'exemple de code suivant montre comment utiliser`Edit-EC2IdFormat`.

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 API plus de détails, consultez la section ModifyIdFormatRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-EC2ImageAttribute`

L'exemple de code suivant montre comment utiliser`Edit-EC2ImageAttribute`.

Outils pour PowerShell

Exemple 1 : Cet exemple met à jour la description du paramètre spécifiéAMI.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Description "New description"
```

Exemple 2 : Cet exemple rend le AMI public (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 le fichier AMI privé (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 API plus de détails, consultez la section ModifyImageAttributeRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-EC2InstanceAttribute`

L'exemple de code suivant montre comment utiliser`Edit-EC2InstanceAttribute`.

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 unVPC. 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 EBS E/S 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 EBS optimisée.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -EbsOptimized $true
```

Exemple 5 : Cet exemple active la vérification source/destination pour l'instance spécifiée. Pour qu'une NAT instance fonctionneNAT, 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 API plus de détails, consultez la section ModifyInstanceAttributeRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-EC2InstanceCreditSpecification`

L'exemple de code suivant montre comment utiliser`Edit-EC2InstanceCreditSpecification`.

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 API plus de détails, consultez la section ModifyInstanceCreditSpecificationRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-EC2NetworkInterfaceAttribute`

L'exemple de code suivant montre comment utiliser`Edit-EC2NetworkInterfaceAttribute`.

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 API plus de détails, consultez la section ModifyNetworkInterfaceAttributeRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-EC2ReservedInstance`

L'exemple de code suivant montre comment utiliser`Edit-EC2ReservedInstance`.

Outils pour PowerShell

Exemple 1 : Cet exemple modifie la zone de disponibilité, le nombre d'instances et la plate-forme 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 API plus de détails, consultez la section ModifyReservedInstancesRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-EC2SnapshotAttribute`

L'exemple de code suivant montre comment utiliser`Edit-EC2SnapshotAttribute`.

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 API plus de détails, consultez la section ModifySnapshotAttributeRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-EC2SpotFleetRequest`

L'exemple de code suivant montre comment utiliser`Edit-EC2SpotFleetRequest`.

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 API plus de détails, consultez la section ModifySpotFleetRequestRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-EC2SubnetAttribute`

L'exemple de code suivant montre comment utiliser`Edit-EC2SubnetAttribute`.

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 API plus de détails, consultez la section ModifySubnetAttributeRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-EC2VolumeAttribute`

L'exemple de code suivant montre comment utiliser`Edit-EC2VolumeAttribute`.

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 API plus de détails, consultez la section ModifyVolumeAttributeRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-EC2VpcAttribute`

L'exemple de code suivant montre comment utiliser`Edit-EC2VpcAttribute`.

Outils pour PowerShell

Exemple 1 : Cet exemple active la prise en charge des DNS noms d'hôtes pour le nom d'hôte spécifiéVPC.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $true
```

Exemple 2 : Cet exemple désactive la prise en charge des DNS noms d'hôtes pour le nom d'hôte spécifié. VPC

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $false
```

Exemple 3 : Cet exemple active la prise en charge de la DNS résolution pour le paramètre spécifiéVPC.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $true
```

Exemple 4 : Cet exemple désactive la prise en charge de la DNS résolution pour le paramètre spécifiéVPC.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $false
```

- Pour API plus de détails, consultez la section ModifyVpcAttributeRéférence des AWS Tools for PowerShell applets de commande.

### `Enable-EC2VgwRoutePropagation`

L'exemple de code suivant montre comment utiliser`Enable-EC2VgwRoutePropagation`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet au paramètre spécifié VGW de propager automatiquement les itinéraires vers la table de routage spécifiée.

```
Enable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- Pour API plus de détails, consultez la section EnableVgwRoutePropagationRéférence des AWS Tools for PowerShell applets de commande.

### `Enable-EC2VolumeIO`

L'exemple de code suivant montre comment utiliser`Enable-EC2VolumeIO`.

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 API plus de détails, consultez la section EnableVolumeIoRéférence des AWS Tools for PowerShell applets de commande.

### `Enable-EC2VpcClassicLink`

L'exemple de code suivant montre comment utiliser`Enable-EC2VpcClassicLink`.

Outils pour PowerShell

Exemple 1 : Cet exemple active VPC vpc-0123456b789b0d12f pour ClassicLink

```
Enable-EC2VpcClassicLink -VpcId vpc-0123456b789b0d12f
```

Sortie :

```
True
```

- Pour API plus de détails, consultez la section EnableVpcClassicLinkRéférence des AWS Tools for PowerShell applets de commande.

### `Enable-EC2VpcClassicLinkDnsSupport`

L'exemple de code suivant montre comment utiliser`Enable-EC2VpcClassicLinkDnsSupport`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet à vpc-0b12d3456a7e8910d de prendre en charge la résolution des noms d'hôte pour DNS ClassicLink

```
Enable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

- Pour API plus de détails, consultez la section EnableVpcClassicLinkDnsSupportRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2AccountAttribute`

L'exemple de code suivant montre comment utiliser`Get-EC2AccountAttribute`.

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 valeur par défautVPC, ou indique « aucune » si aucune valeur par défaut n'existe VPC 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 API plus de détails, consultez la section DescribeAccountAttributesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2Address`

L'exemple de code suivant montre comment utiliser`Get-EC2Address`.

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'unVPC. 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 de 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 tous les Elastic IPs avec son identifiant d'allocation, son identifiant d'association et ses identifiants d'instance

```
Get-EC2Address -Region eu-west-1 | Select-Object InstanceId, AssociationId,
 AllocationId, PublicIp
```

Sortie :

```
InstanceId          AssociationId              AllocationId               PublicIp
----------          -------------              ------------               --------
                                               eipalloc-012e3b456789e1fad
 17.212.120.178
i-0c123dfd3415bac67 eipassoc-0e123456bb7890bdb eipalloc-01cd23ebf45f7890c
 17.212.124.77
                                               eipalloc-012345678eeabcfad
 17.212.225.7
i-0123d405c67e89a0c eipassoc-0c123b456783966ba eipalloc-0123cdd456a8f7892
 37.216.52.173
i-0f1bf2f34c5678d09 eipassoc-0e12934568a952d96 eipalloc-0e1c23e4d5e6789e4
 37.218.222.278
i-012e3cb4df567e8aa eipassoc-0d1b2fa4d67d03810 eipalloc-0123f456f78a01b58
 37.210.82.27
i-0123bcf4b567890e1 eipassoc-01d2345f678903fb1 eipalloc-0e1db23cfef5c45c7
 37.215.222.270
```

Exemple 8 : Cet exemple extrait la liste des adresses EC2 IP 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 API plus de détails, consultez la section DescribeAddressesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2AvailabilityZone`

L'exemple de code suivant montre comment utiliser`Get-EC2AvailabilityZone`.

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 API plus de détails, consultez la section DescribeAvailabilityZonesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2BundleTask`

L'exemple de code suivant montre comment utiliser`Get-EC2BundleTask`.

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 API plus de détails, consultez la section DescribeBundleTasksRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2CapacityReservation`

L'exemple de code suivant montre comment utiliser`Get-EC2CapacityReservation`.

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 API plus de détails, consultez la section DescribeCapacityReservationsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2ConsoleOutput`

L'exemple de code suivant montre comment utiliser`Get-EC2ConsoleOutput`.

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 API plus de détails, consultez la section GetConsoleOutputRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2CustomerGateway`

L'exemple de code suivant montre comment utiliser`Get-EC2CustomerGateway`.

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 API plus de détails, consultez la section DescribeCustomerGatewaysRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2DhcpOption`

L'exemple de code suivant montre comment utiliser`Get-EC2DhcpOption`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie vos ensembles DHCP d'options.

```
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 DHCP d'options 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 API plus de détails, consultez la section DescribeDhcpOptionsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2FlowLog`

L'exemple de code suivant montre comment utiliser`Get-EC2FlowLog`.

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                : fl-01b2e3d45f67f8901
FlowLogStatus            : ACTIVE
LogDestination           : arn:aws:s3:::my-bucket-dd-tata
LogDestinationType       : s3
LogGroupName             :
ResourceId               : eni-01d2dda3456b7e890
TrafficType              : ALL
```

- Pour API plus de détails, consultez la section DescribeFlowLogsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2Host`

L'exemple de code suivant montre comment utiliser`Get-EC2Host`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie les détails de l'EC2hôte

```
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 API plus de détails, consultez la section DescribeHostsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2HostReservationOffering`

L'exemple de code suivant montre comment utiliser`Get-EC2HostReservationOffering`.

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 API plus de détails, consultez la section DescribeHostReservationOfferingsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2HostReservationPurchasePreview`

L'exemple de code suivant montre comment utiliser`Get-EC2HostReservationPurchasePreview`.

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 API plus de détails, consultez la section GetHostReservationPurchasePreviewRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2IdFormat`

L'exemple de code suivant montre comment utiliser`Get-EC2IdFormat`.

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 extensionsIDs.

```
Get-EC2IdFormat
```

Sortie :

```
Resource       UseLongIds
--------       ----------
reservation    False
instance       False
```

- Pour API plus de détails, consultez la section DescribeIdFormatRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2IdentityIdFormat`

L'exemple de code suivant montre comment utiliser`Get-EC2IdentityIdFormat`.

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 API plus de détails, consultez la section DescribeIdentityIdFormatRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2Image`

L'exemple de code suivant montre comment utiliser`Get-EC2Image`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit le paramètre spécifiéAMI.

```
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 le produit AMIs que vous possédez.

```
Get-EC2Image -owner self
```

Exemple 3 : Cet exemple décrit le public AMIs qui exécute Microsoft Windows Server.

```
Get-EC2Image -Filter @{ Name="platform"; Values="windows" }
```

Exemple 4 : Cet exemple décrit tous les publics de la AMIs région « us-west-2 ».

```
Get-EC2Image -Region us-west-2
```

- Pour API plus de détails, consultez la section DescribeImagesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2ImageAttribute`

L'exemple de code suivant montre comment utiliser`Get-EC2ImageAttribute`.

Outils pour PowerShell

Exemple 1 : Cet exemple obtient la description du paramètre spécifiéAMI.

```
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 le fichier spécifiéAMI.

```
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 API plus de détails, consultez la section DescribeImageAttributeRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2ImageByName`

L'exemple de code suivant montre comment utiliser`Get-EC2ImageByName`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit l'ensemble complet des noms de filtres actuellement pris en charge.

```
Get-EC2ImageByName
```

Sortie :

```
WINDOWS_2016_BASE
```

```
WINDOWS_2016_NANO
WINDOWS_2016_CORE
WINDOWS_2016_CONTAINER
WINDOWS_2016_SQL_SERVER_ENTERPRISE_2016
WINDOWS_2016_SQL_SERVER_STANDARD_2016
WINDOWS_2016_SQL_SERVER_WEB_2016
WINDOWS_2016_SQL_SERVER_EXPRESS_2016
WINDOWS_2012R2_BASE
WINDOWS_2012R2_CORE
WINDOWS_2012R2_SQL_SERVER_EXPRESS_2016
WINDOWS_2012R2_SQL_SERVER_STANDARD_2016
WINDOWS_2012R2_SQL_SERVER_WEB_2016
WINDOWS_2012R2_SQL_SERVER_EXPRESS_2014
WINDOWS_2012R2_SQL_SERVER_STANDARD_2014
WINDOWS_2012R2_SQL_SERVER_WEB_2014
WINDOWS_2012_BASE
WINDOWS_2012_SQL_SERVER_EXPRESS_2014
WINDOWS_2012_SQL_SERVER_STANDARD_2014
WINDOWS_2012_SQL_SERVER_WEB_2014
WINDOWS_2012_SQL_SERVER_EXPRESS_2012
WINDOWS_2012_SQL_SERVER_STANDARD_2012
WINDOWS_2012_SQL_SERVER_WEB_2012
WINDOWS_2012_SQL_SERVER_EXPRESS_2008
WINDOWS_2012_SQL_SERVER_STANDARD_2008
WINDOWS_2012_SQL_SERVER_WEB_2008
WINDOWS_2008R2_BASE
WINDOWS_2008R2_SQL_SERVER_EXPRESS_2012
WINDOWS_2008R2_SQL_SERVER_STANDARD_2012
WINDOWS_2008R2_SQL_SERVER_WEB_2012
WINDOWS_2008R2_SQL_SERVER_EXPRESS_2008
WINDOWS_2008R2_SQL_SERVER_STANDARD_2008
WINDOWS_2008R2_SQL_SERVER_WEB_2008
WINDOWS_2008RTM_BASE
WINDOWS_2008RTM_SQL_SERVER_EXPRESS_2008
WINDOWS_2008RTM_SQL_SERVER_STANDARD_2008
WINDOWS_2008_BEANSTALK_IIS75
WINDOWS_2012_BEANSTALK_IIS8
VPC_NAT
```

Exemple 2 : Cet exemple décrit le paramètre spécifiéAMI. L'utilisation de cette commande pour localiser un AMI est utile car AWS chaque mois, un nouveau Windows est publié AMIs avec les dernières mises à jour. Vous pouvez spécifier le « ImageId » pour New-EC2Instance lancer une instance en utilisant le courant AMI pour le filtre spécifié.

```
Get-EC2ImageByName -Names WINDOWS_2016_BASE
```

Sortie :

```
Architecture        : x86_64
BlockDeviceMappings : {/dev/sda1, xvdca, xvdcb, xvdcc...}
CreationDate        : yyyy.mm.ddThh:mm:ss.000Z
Description         : Microsoft Windows Server 2016 with Desktop Experience Locale
 English AMI provided by Amazon
Hypervisor          : xen
ImageId             : ami-xxxxxxxx
ImageLocation       : amazon/Windows_Server-2016-English-Full-Base-yyyy.mm.dd
ImageOwnerAlias     : amazon
ImageType           : machine
KernelId            :
Name                : Windows_Server-2016-English-Full-Base-yyyy.mm.dd
OwnerId             : 801119661308
Platform            : Windows
ProductCodes        : {}
Public              : True
RamdiskId           :
RootDeviceName      : /dev/sda1
RootDeviceType      : ebs
SriovNetSupport     : simple
State               : available
StateReason         :
Tags                : {}
VirtualizationType  : hvm
```

- Pour API plus de détails, consultez la section Get-EC2ImageByNameRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2ImportImageTask`

L'exemple de code suivant montre comment utiliser`Get-EC2ImportImageTask`.

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 API plus de détails, consultez la section DescribeImportImageTasksRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2ImportSnapshotTask`

L'exemple de code suivant montre comment utiliser`Get-EC2ImportSnapshotTask`.

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 API plus de détails, consultez la section DescribeImportSnapshotTasksRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2Instance`

L'exemple de code suivant montre comment utiliser`Get-EC2Instance`.

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           : TleEy1448154045270
EbsOptimized          : False
Hypervisor            : xen
IamInstanceProfile    : Amazon.EC2.Model.IamInstanceProfile
ImageId               : ami-12345678
InstanceId            : i-12345678
InstanceLifecycle     :
InstanceType          : t2.micro
KernelId              :
KeyName               : my-key-pair
LaunchTime            : 12/4/2015 4:44:40 PM
Monitoring            : Amazon.EC2.Model.Monitoring
NetworkInterfaces     : {ip-10-0-2-172.us-west-2.compute.internal}
Placement             : Amazon.EC2.Model.Placement
Platform              : Windows
PrivateDnsName        : ip-10-0-2-172.us-west-2.compute.internal
PrivateIpAddress      : 10.0.2.172
ProductCodes          : {}
PublicDnsName         :
PublicIpAddress       :
RamdiskId             :
RootDeviceName        : /dev/sda1
RootDeviceType        : ebs
SecurityGroups        : {default}
SourceDestCheck       : True
SpotInstanceRequestId :
SriovNetSupport       :
State                 : Amazon.EC2.Model.InstanceState
```

```
StateReason           :
StateTransitionReason :
SubnetId              : subnet-12345678
Tags                  : {Name}
VirtualizationType    : hvm
VpcId                 : vpc-12345678
```

Exemple 2 : Cet exemple décrit toutes vos instances dans la région actuelle, regroupées par réservation. Pour voir les détails des instances, étendez la collection Instances au sein de 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 EC2 instances dans un sous-réseau spécifique d'unVPC.

```
(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 API plus de détails, consultez la section DescribeInstancesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2InstanceAttribute`

L'exemple de code suivant montre comment utiliser`Get-EC2InstanceAttribute`.

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 EBS l'optimisation 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 API plus de détails, consultez la section DescribeInstanceAttributeRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2InstanceMetadata`

L'exemple de code suivant montre comment utiliser`Get-EC2InstanceMetadata`.

Exemple 1 : Répertorie les catégories de métadonnées d'instance disponibles qui peuvent être interrogées.

```
Get-EC2InstanceMetadata -ListCategory
```

Sortie :

```
AmiId
LaunchIndex
ManifestPath
AncestorAmiId
BlockDeviceMapping
InstanceId
InstanceType
LocalHostname
LocalIpv4
KernelId
AvailabilityZone
ProductCode
PublicHostname
PublicIpv4
PublicKey
RamdiskId
Region
ReservationId
SecurityGroup
UserData
InstanceMonitoring
IdentityDocument
IdentitySignature
IdentityPkcs7
```

Exemple 2 : renvoie l'identifiant de l'Amazon Machine Image (AMI) qui a été utilisée pour lancer l'instance.

```
Get-EC2InstanceMetadata -Category AmiId
```

Sortie :

```
ami-b2e756ca
```

Exemple 3 : Cet exemple interroge le document d'identité au JSON format -formaté pour l'instance.

```
Get-EC2InstanceMetadata -Category IdentityDocument
{
  "availabilityZone" : "us-west-2a",
  "devpayProductCodes" : null,
  "marketplaceProductCodes" : null,
  "version" : "2017-09-30",
  "instanceId" : "i-01ed50f7e2607f09e",
  "billingProducts" : [ "bp-6ba54002" ],
  "instanceType" : "t2.small",
  "pendingTime" : "2018-03-07T16:26:04Z",
  "imageId" : "ami-b2e756ca",
  "privateIp" : "10.0.0.171",
  "accountId" : "111122223333",
  "architecture" : "x86_64",
  "kernelId" : null,
  "ramdiskId" : null,
  "region" : "us-west-2"
}
```

Exemple 4 : Cet exemple utilise une requête de chemin pour obtenir les macs de l'interface réseau pour l'instance.

```
Get-EC2InstanceMetadata -Path "/network/interfaces/macs"
```

Sortie :

```
02:80:7f:ef:4c:e0/
```

Exemple 5 : Si un IAM rôle est associé à l'instance, renvoie des informations sur la dernière mise à jour du profil de l'instance, y compris la LastUpdated date de l'instance InstanceProfileArn, et InstanceProfileId.

```
Get-EC2InstanceMetadata -Path "/iam/info"
```

Sortie :

```
{
  "Code" : "Success",
  "LastUpdated" : "2018-03-08T03:38:40Z",
  "InstanceProfileArn" : "arn:aws:iam::111122223333:instance-profile/
MyLaunchRole_Profile",
  "InstanceProfileId" : "AIPAI4...WVK2RW"
}
```

- Pour API plus de détails, consultez la section Get-EC2InstanceMetadataRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2InstanceStatus`

L'exemple de code suivant montre comment utiliser`Get-EC2InstanceStatus`.

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 API plus de détails, consultez la section DescribeInstanceStatusRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2InternetGateway`

L'exemple de code suivant montre comment utiliser`Get-EC2InternetGateway`.

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 API plus de détails, consultez la section DescribeInternetGatewaysRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2KeyPair`

L'exemple de code suivant montre comment utiliser`Get-EC2KeyPair`.

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 API plus de détails, consultez la section DescribeKeyPairsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2NetworkAcl`

L'exemple de code suivant montre comment utiliser`Get-EC2NetworkAcl`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit le réseau spécifiéACL.

```
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 le réseau spécifiéACL.

```
(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 l'ensemble de votre réseauACLs.

```
Get-EC2NetworkAcl
```

- Pour API plus de détails, consultez la section DescribeNetworkAclsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2NetworkInterface`

L'exemple de code suivant montre comment utiliser`Get-EC2NetworkInterface`.

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 API plus de détails, consultez la section DescribeNetworkInterfacesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2NetworkInterfaceAttribute`

L'exemple de code suivant montre comment utiliser`Get-EC2NetworkInterfaceAttribute`.

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 API plus de détails, consultez la section DescribeNetworkInterfaceAttributeRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2PasswordData`

L'exemple de code suivant montre comment utiliser`Get-EC2PasswordData`.

Outils pour PowerShell

Exemple 1 : Cet exemple déchiffre le mot de passe EC2 attribué par Amazon 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 API plus de détails, consultez la section GetPasswordDataRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2PlacementGroup`

L'exemple de code suivant montre comment utiliser`Get-EC2PlacementGroup`.

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 API plus de détails, consultez la section DescribePlacementGroupsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2PrefixList`

L'exemple de code suivant montre comment utiliser`Get-EC2PrefixList`.

Outils pour PowerShell

Exemple 1 : Cet exemple extrait le fichier disponible Services AWS 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 API plus de détails, consultez la section DescribePrefixListsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2Region`

L'exemple de code suivant montre comment utiliser`Get-EC2Region`.

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 API plus de détails, consultez la section DescribeRegionsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2RouteTable`

L'exemple de code suivant montre comment utiliser`Get-EC2RouteTable`.

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 les données spécifiéesVPC.

```
Get-EC2RouteTable -Filter @{ Name="vpc-id"; Values="vpc-1a2b3c4d" }
```

Sortie :

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

- Pour API plus de détails, consultez la section DescribeRouteTablesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2ScheduledInstance`

L'exemple de code suivant montre comment utiliser`Get-EC2ScheduledInstance`.

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 API plus de détails, consultez la section DescribeScheduledInstancesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2ScheduledInstanceAvailability`

L'exemple de code suivant montre comment utiliser`Get-EC2ScheduledInstanceAvailability`.

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 API plus de détails, consultez la section DescribeScheduledInstanceAvailabilityRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2SecurityGroup`

L'exemple de code suivant montre comment utiliser`Get-EC2SecurityGroup`.

Exemple 1 : Cet exemple décrit le groupe de sécurité spécifié pour unVPC. 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 API plus de détails, consultez la section DescribeSecurityGroupsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2Snapshot`

L'exemple de code suivant montre comment utiliser`Get-EC2Snapshot`.

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 API plus de détails, consultez la section DescribeSnapshotsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2SnapshotAttribute`

L'exemple de code suivant montre comment utiliser`Get-EC2SnapshotAttribute`.

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 API plus de détails, consultez la section DescribeSnapshotAttributeRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2SpotDatafeedSubscription`

L'exemple de code suivant montre comment utiliser`Get-EC2SpotDatafeedSubscription`.

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 API plus de détails, consultez la section DescribeSpotDatafeedSubscriptionRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2SpotFleetInstance`

L'exemple de code suivant montre comment utiliser`Get-EC2SpotFleetInstance`.

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 API plus de détails, consultez la section DescribeSpotFleetInstancesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2SpotFleetRequest`

L'exemple de code suivant montre comment utiliser`Get-EC2SpotFleetRequest`.

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 API plus de détails, consultez la section DescribeSpotFleetRequestsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2SpotFleetRequestHistory`

L'exemple de code suivant montre comment utiliser`Get-EC2SpotFleetRequestHistory`.

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 API plus de détails, consultez la section DescribeSpotFleetRequestHistoryRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2SpotInstanceRequest`

L'exemple de code suivant montre comment utiliser`Get-EC2SpotInstanceRequest`.

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 API plus de détails, consultez la section DescribeSpotInstanceRequestsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2SpotPriceHistory`

L'exemple de code suivant montre comment utiliser`Get-EC2SpotPriceHistory`.

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 API plus de détails, consultez la section DescribeSpotPriceHistoryRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2Subnet`

L'exemple de code suivant montre comment utiliser`Get-EC2Subnet`.

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 API plus de détails, consultez la section DescribeSubnetsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2Tag`

L'exemple de code suivant montre comment utiliser`Get-EC2Tag`.

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 API plus de détails, consultez la section DescribeTagsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2Volume`

L'exemple de code suivant montre comment utiliser`Get-EC2Volume`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit le EBS volume 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 EBS volumes 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 EBS volumes.

```
Get-EC2Volume
```

- Pour API plus de détails, consultez la section DescribeVolumesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2VolumeAttribute`

L'exemple de code suivant montre comment utiliser`Get-EC2VolumeAttribute`.

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 API plus de détails, consultez la section DescribeVolumeAttributeRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2VolumeStatus`

L'exemple de code suivant montre comment utiliser`Get-EC2VolumeStatus`.

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 API plus de détails, consultez la section DescribeVolumeStatusRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2Vpc`

L'exemple de code suivant montre comment utiliser`Get-EC2Vpc`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit le paramètre spécifiéVPC.

```
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 la valeur par défaut VPC (il ne peut y en avoir qu'une par région). Si votre compte prend en charge le EC2 mode -Classic dans cette région, il n'existe aucune option par défautVPC.

```
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 ceux VPCs qui correspondent au filtre spécifié (c'est-à-dire ceux dont l'état correspond à la valeur « 10.0.0.0/16 » et CIDR qui sont dans l'état « disponible »).

```
Get-EC2Vpc -Filter @{Name="cidr";
 Values="10.0.0.0/16"},@{Name="state";Values="available"}
```

Exemple 4 : Cet exemple décrit tous vosVPCs.

```
Get-EC2Vpc
```

- Pour API plus de détails, consultez la section DescribeVpcsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2VpcAttribute`

L'exemple de code suivant montre comment utiliser`Get-EC2VpcAttribute`.

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 API plus de détails, consultez la section DescribeVpcAttributeRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2VpcClassicLink`

L'exemple de code suivant montre comment utiliser`Get-EC2VpcClassicLink`.

Outils pour PowerShell

Exemple 1 : L'exemple ci-dessus renvoie tous les VPCs 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 API plus de détails, consultez la section DescribeVpcClassicLinkRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2VpcClassicLinkDnsSupport`

L'exemple de code suivant montre comment utiliser`Get-EC2VpcClassicLinkDnsSupport`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit le statut de ClassicLink DNS support de la VPCs région eu-west-1

```
Get-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

Sortie :

```
ClassicLinkDnsSupported VpcId
----------------------- -----
False                   vpc-0b12d3456a7e8910d
False                   vpc-12cf3b4f
```

- Pour API plus de détails, consultez la section DescribeVpcClassicLinkDnsSupportRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2VpcEndpoint`

L'exemple de code suivant montre comment utiliser`Get-EC2VpcEndpoint`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit un ou plusieurs de vos VPC points de terminaison pour la région eu-west-1. Il dirige ensuite la sortie vers la commande suivante, qui sélectionne la VpcEndpointId propriété et renvoie l'VPCID 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.ssmmessages
          True
```

Exemple 3 : Cet exemple exporte le document de politique pour le point de VPC terminaison 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 API plus de détails, consultez la section DescribeVpcEndpointsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2VpcEndpointService`

L'exemple de code suivant montre comment utiliser`Get-EC2VpcEndpointService`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit le service de point de EC2 VPC terminaison 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 EC2 VPC Endpoint 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 API plus de détails, consultez la section DescribeVpcEndpointServicesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2VpnConnection`

L'exemple de code suivant montre comment utiliser`Get-EC2VpnConnection`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit la VPN connexion 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 VPN connexion 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 VPN connexions.

```
Get-EC2VpnConnection
```

- Pour API plus de détails, consultez la section DescribeVpnConnectionsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EC2VpnGateway`

L'exemple de code suivant montre comment utiliser`Get-EC2VpnGateway`.

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 API plus de détails, consultez la section DescribeVpnGatewaysRéférence des AWS Tools for PowerShell applets de commande.

### `Grant-EC2SecurityGroupEgress`

L'exemple de code suivant montre comment utiliser`Grant-EC2SecurityGroupEgress`.

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 TCP port 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 TCP port 80.

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

Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission
 @( @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; UserIdGroupPairs=$ug } )
```

- Pour API plus de détails, consultez la section AuthorizeSecurityGroupEgressRéférence des AWS Tools for PowerShell applets de commande.

### `Grant-EC2SecurityGroupIngress`

L'exemple de code suivant montre comment utiliser`Grant-EC2SecurityGroupIngress`.

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 SSH (port 22) et RDC (port 3389). Notez que vous devez identifier les groupes de sécurité pour EC2 : VPC en utilisant l'ID du groupe de sécurité et non le 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 SSH (port 22) et 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 au TCP port 8081 l'accès 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 22 avec une description. TCP

```
$IpRange = New-Object -TypeName Amazon.EC2.Model.IpRange
$IpRange.CidrIp = "5.5.5.5/32"
$IpRange.Description = "SSH from Office"
$IpPermission = New-Object Amazon.EC2.Model.IpPermission
$IpPermission.IpProtocol = "tcp"
$IpPermission.ToPort = 22
$IpPermission.FromPort = 22
```

```
$IpPermission.Ipv4Ranges = $IpRange
Grant-EC2SecurityGroupIngress -GroupId sg-1234abcd -IpPermission $IpPermission
```

- Pour API plus de détails, consultez la section AuthorizeSecurityGroupIngressRéférence des AWS Tools for PowerShell applets de commande.

### `Import-EC2Image`

L'exemple de code suivant montre comment utiliser`Import-EC2Image`.

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'idempuissance. 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 à EC2 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 = "amzn-s3-demo-bucket"
$container.UserBucket.S3Key = "Win_2008_Server_Standard_SP2_64-bit-disk1.vmdk"

$parms = @{
    "ClientToken"="idempotencyToken"
    "Description"="Windows 2008 Standard Image Import"
    "Platform"="Windows"
    "LicenseType"="AWS"
}

Import-EC2Image -DiskContainer $container @parms
```

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 API plus de détails, consultez la section ImportImageRéférence des AWS Tools for PowerShell applets de commande.

### `Import-EC2KeyPair`

L'exemple de code suivant montre comment utiliser`Import-EC2KeyPair`.

Outils pour PowerShell

Exemple 1 : Cet exemple importe une clé publique dansEC2. La première ligne stocke le contenu du fichier de clé publique (*.pub) dans la variable**`$publickey`**. Ensuite, l'exemple convertit le UTF8 format du fichier de clé publique en une chaîne codée en Base64 et stocke la chaîne convertie dans la variable. **`$pkbase64`** Dans la dernière ligne, la clé publique convertie est importée dansEC2. 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 API plus de détails, consultez la section ImportKeyPairRéférence des AWS Tools for PowerShell applets de commande.

### `Import-EC2Snapshot`

L'exemple de code suivant montre comment utiliser`Import-EC2Snapshot`.

Outils pour PowerShell

Exemple 1 : Cet exemple importe une image de disque de machine virtuelle au format VMDK « » dans un EBS instantané Amazon. L'exemple nécessite un rôle de service d'importation de machine virtuelle portant le nom par défaut « vmimport », avec une politique autorisant Amazon à EC2 accéder au compartiment spécifié, comme expliqué dans la **`VM Import Prequisites`** rubrique http://docs.aws.amazon.com/ AWSEC2 WindowsGuide /latest//.html. VMImportPrerequisites Pour utiliser un rôle personnalisé, spécifiez le nom du rôle à l'aide du **`-RoleName`** paramètre.

```
$parms = @{
    "ClientToken"="idempotencyToken"
    "Description"="Disk Image Import"
    "DiskContainer_Description" = "Data disk"
    "DiskContainer_Format" = "VMDK"
    "DiskContainer_S3Bucket" = "amzn-s3-demo-bucket"
    "DiskContainer_S3Key" = "datadiskimage.vmdk"
}

Import-EC2Snapshot @parms
```

Sortie :

```
Description            ImportTaskId               SnapshotTaskDetail

-----------------      --------------------       ------------------

Disk Image Import      import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
```

- Pour API plus de détails, consultez la section ImportSnapshotRéférence des AWS Tools for PowerShell applets de commande.

### `Move-EC2AddressToVpc`

L'exemple de code suivant montre comment utiliser`Move-EC2AddressToVpc`.

Outils pour PowerShell

Exemple 1 : Cet exemple déplace une EC2 instance dont l'adresse IP publique est 12.345.67.89 vers la VPC plate-forme EC2 - 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-EC2AddressToVpcapplet 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 API plus de détails, consultez la section MoveAddressToVpcRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2Address`

L'exemple de code suivant montre comment utiliser`New-EC2Address`.

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 API plus de détails, consultez la section AllocateAddressRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2CustomerGateway`

L'exemple de code suivant montre comment utiliser`New-EC2CustomerGateway`.

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 API plus de détails, consultez la section CreateCustomerGatewayRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2DhcpOption`

L'exemple de code suivant montre comment utiliser`New-EC2DhcpOption`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée l'ensemble d'DHCPoptions 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 DHCP option.

```
$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 API plus de détails, consultez la section CreateDhcpOptionsRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2FlowLog`

L'exemple de code suivant montre comment utiliser`New-EC2FlowLog`.

Outils pour PowerShell

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

```
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//VUKl5EU6cF7gQLOxcqNefvjeTGw= {fl-012fc34eed5678c9d} {}
```

- Pour API plus de détails, consultez la section CreateFlowLogsRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2Host`

L'exemple de code suivant montre comment utiliser`New-EC2Host`.

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 API plus de détails, consultez la section AllocateHostsRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2HostReservation`

L'exemple de code suivant montre comment utiliser`New-EC2HostReservation`.

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 API plus de détails, consultez la section PurchaseHostReservationRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2Image`

L'exemple de code suivant montre comment utiliser`New-EC2Image`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un 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 un 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 un fichier AMI avec trois volumes. Le premier volume est basé sur un EBS instantané Amazon. 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 API plus de détails, consultez la section CreateImageRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2Instance`

L'exemple de code suivant montre comment utiliser`New-EC2Instance`.

Outils pour PowerShell

Exemple 1 : Cet exemple lance une seule instance du paramètre spécifié AMI dans EC2 -Classic ou une instance par défautVPC.

```
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 seule instance du paramètre spécifié AMI dans unVPC.

```
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 EBS volume 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'un des Windows actuelsAMIs, obtenez son AMI identifiant à l'aide deGet-EC2ImageByName. Cet exemple lance une instance à partir de la base actuelle AMI 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 API plus de détails, consultez la section RunInstancesRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2InstanceExportTask`

L'exemple de code suivant montre comment utiliser`New-EC2InstanceExportTask`.

Outils pour PowerShell

Exemple 1 : Cet exemple exporte une instance arrêtée**`i-0800b00a00EXAMPLE`**, sous forme de disque dur virtuel (VHD) vers le compartiment S3**`testbucket-export-instances-2019`**. 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 "amzn-s3-demo-bucket" -
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 API plus de détails, consultez la section CreateInstanceExportTaskRéférence des AWS Tools for PowerShell applets de commande.

### New-EC2InternetGateway

L'exemple de code suivant montre comment utiliserNew-EC2InternetGateway.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une passerelle Internet.

```
New-EC2InternetGateway
```

Sortie :

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

- Pour API plus de détails, consultez la section CreateInternetGatewayRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2KeyPair`

L'exemple de code suivant montre comment utiliser`New-EC2KeyPair`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une paire de clés et capture la clé RSA privée PEM codée 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 Créer, afficher et supprimer des paires de EC2 clés Amazon (https://docs.aws.amazon.com/cli/latest/userguide/ cli-services-ec2-keypairs.html) dans le guide de l'utilisateur de l'interface de AWS ligne de commande.

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

- Pour API plus de détails, consultez la section CreateKeyPairRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2NetworkAcl`

L'exemple de code suivant montre comment utiliser`New-EC2NetworkAcl`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un réseau ACL pour le spécifiéVPC.

```
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 API plus de détails, consultez la section CreateNetworkAclRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2NetworkAclEntry`

L'exemple de code suivant montre comment utiliser`New-EC2NetworkAclEntry`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une entrée pour le réseau spécifiéACL. La règle autorise le trafic entrant en provenance de n'importe où (0.0.0.0/0) sur le UDP port 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 API plus de détails, consultez la section CreateNetworkAclEntryRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2NetworkInterface`

L'exemple de code suivant montre comment utiliser`New-EC2NetworkInterface`.

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
PrivateIpAddresses : {}
```

```
RequesterId        :
RequesterManaged   : False
SourceDestCheck    : True
Status             : pending
SubnetId           : subnet-1a2b3c4d
TagSet             : {}
VpcId              : vpc-12345678
```

- Pour API plus de détails, consultez la section CreateNetworkInterfaceRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2PlacementGroup`

L'exemple de code suivant montre comment utiliser`New-EC2PlacementGroup`.

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 API plus de détails, consultez la section CreatePlacementGroupRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2Route`

L'exemple de code suivant montre comment utiliser`New-EC2Route`.

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 API plus de détails, consultez la section CreateRouteRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2RouteTable`

L'exemple de code suivant montre comment utiliser`New-EC2RouteTable`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une table de routage pour le paramètre spécifiéVPC.

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

Sortie :

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

- Pour API plus de détails, consultez la section CreateRouteTableRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2ScheduledInstance`

L'exemple de code suivant montre comment utiliser`New-EC2ScheduledInstance`.

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 API plus de détails, consultez la section RunScheduledInstancesRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2ScheduledInstancePurchase`

L'exemple de code suivant montre comment utiliser`New-EC2ScheduledInstancePurchase`.

Outils pour PowerShell

Exemple 1 : Cet exemple achète une instance planifiée.

```
$request = New-Object Amazon.EC2.Model.PurchaseRequest
$request.InstanceCount = 1
$request.PurchaseToken = "eyJ2IjoiMSIsInMiOjEsImMiOi..."
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 API plus de détails, consultez la section PurchaseScheduledInstancesRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2SecurityGroup`

L'exemple de code suivant montre comment utiliser`New-EC2SecurityGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un groupe de sécurité pour le groupe spécifiéVPC.

```
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 -Classic.

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

Sortie :

```
sg-45678901
```

- Pour API plus de détails, consultez la section CreateSecurityGroupRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2Snapshot`

L'exemple de code suivant montre comment utiliser`New-EC2Snapshot`.

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 API plus de détails, consultez la section CreateSnapshotRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2SpotDatafeedSubscription`

L'exemple de code suivant montre comment utiliser`New-EC2SpotDatafeedSubscription`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un flux de données d'instance Spot.

```
New-EC2SpotDatafeedSubscription -Bucket amzn-s3-demo-bucket -Prefix spotdata
```

Sortie :

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

- Pour API plus de détails, consultez la section CreateSpotDatafeedSubscriptionRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2Subnet`

L'exemple de code suivant montre comment utiliser`New-EC2Subnet`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un sous-réseau avec le paramètre spécifiéCIDR.

```
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 API plus de détails, consultez la section CreateSubnetRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2Tag`

L'exemple de code suivant montre comment utiliser`New-EC2Tag`.

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 API plus de détails, consultez la section CreateTagsRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2Volume`

L'exemple de code suivant montre comment utiliser`New-EC2Volume`.

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 API plus de détails, consultez la section CreateVolumeRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2Vpc`

L'exemple de code suivant montre comment utiliser`New-EC2Vpc`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un VPC avec le paramètre spécifiéCIDR. Amazon crée VPC également les éléments suivants pour VPC : un ensemble d'DHCPoptions par défaut, une table de routage principale et un réseau par défautACL.

```
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 API plus de détails, consultez la section CreateVpcRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2VpcEndpoint`

L'exemple de code suivant montre comment utiliser`New-EC2VpcEndpoint`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un nouveau VPC point de terminaison pour le service com.amazonaws.eu-west-1.s3 dans le fichier vpc-0fc1ff23f45b678eb VPC

```
New-EC2VpcEndpoint -ServiceName com.amazonaws.eu-west-1.s3 -VpcId
 vpc-0fc1ff23f45b678eb
```

Sortie :

```
ClientToken VpcEndpoint
----------- -----------
            Amazon.EC2.Model.VpcEndpoint
```

- Pour API plus de détails, consultez la section CreateVpcEndpointRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2VpnConnection`

L'exemple de code suivant montre comment utiliser`New-EC2VpnConnection`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une VPN connexion entre la passerelle privée virtuelle spécifiée et la passerelle client spécifiée. La sortie inclut les informations de configuration dont votre administrateur réseau a besoin, sous XML forme de format.

```
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 VPN connexion 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 VPN connexion, 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 API plus de détails, consultez la section CreateVpnConnectionRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2VpnConnectionRoute`

L'exemple de code suivant montre comment utiliser`New-EC2VpnConnectionRoute`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée la route statique spécifiée pour la VPN connexion spécifiée.

```
New-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock
 11.12.0.0/16
```

- Pour API plus de détails, consultez la section CreateVpnConnectionRouteRéférence des AWS Tools for PowerShell applets de commande.

### `New-EC2VpnGateway`

L'exemple de code suivant montre comment utiliser`New-EC2VpnGateway`.

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 API plus de détails, consultez la section CreateVpnGatewayRéférence des AWS Tools for PowerShell applets de commande.

### `Register-EC2Address`

L'exemple de code suivant montre comment utiliser`Register-EC2Address`.

Outils pour PowerShell

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

```
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 API plus de détails, consultez la section AssociateAddressRéférence des AWS Tools for PowerShell applets de commande.

### `Register-EC2DhcpOption`

L'exemple de code suivant montre comment utiliser`Register-EC2DhcpOption`.

Outils pour PowerShell

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

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

Exemple 2 : Cet exemple associe les DHCP options définies par défaut aux options spécifiéesVPC.

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

- Pour API plus de détails, consultez la section AssociateDhcpOptionsRéférence des AWS Tools for PowerShell applets de commande.

### `Register-EC2Image`

L'exemple de code suivant montre comment utiliser`Register-EC2Image`.

Outils pour PowerShell

Exemple 1 : Cet exemple enregistre un AMI en utilisant le fichier manifeste spécifié dans Amazon S3.

```
Register-EC2Image -ImageLocation amzn-s3-demo-bucket/my-web-server-ami/
image.manifest.xml -Name my-web-server-ami
```

- Pour API plus de détails, consultez la section RegisterImageRéférence des AWS Tools for PowerShell applets de commande.

### `Register-EC2PrivateIpAddress`

L'exemple de code suivant montre comment utiliser`Register-EC2PrivateIpAddress`.

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 API plus de détails, consultez la section AssignPrivateIpAddressesRéférence des AWS Tools for PowerShell applets de commande.

### `Register-EC2RouteTable`

L'exemple de code suivant montre comment utiliser`Register-EC2RouteTable`.

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 API plus de détails, consultez la section AssociateRouteTableRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EC2Address`

L'exemple de code suivant montre comment utiliser`Remove-EC2Address`.

Outils pour PowerShell

Exemple 1 : Cet exemple libère l'adresse IP élastique spécifiée pour les instances d'unVPC.

```
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 API plus de détails, consultez la section ReleaseAddressRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EC2CapacityReservation`

L'exemple de code suivant montre comment utiliser`Remove-EC2CapacityReservation`.

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 API plus de détails, consultez la section CancelCapacityReservationRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EC2CustomerGateway`

L'exemple de code suivant montre comment utiliser`Remove-EC2CustomerGateway`.

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 API plus de détails, consultez la section DeleteCustomerGatewayRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EC2DhcpOption`

L'exemple de code suivant montre comment utiliser`Remove-EC2DhcpOption`.

Outils pour PowerShell

Exemple 1 : cet exemple supprime le jeu DHCP d'options 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 API plus de détails, consultez la section DeleteDhcpOptionsRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EC2FlowLog`

L'exemple de code suivant montre comment utiliser`Remove-EC2FlowLog`.

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 API plus de détails, consultez la section DeleteFlowLogsRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EC2Host`

L'exemple de code suivant montre comment utiliser`Remove-EC2Host`.

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 API plus de détails, consultez la section ReleaseHostsRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EC2Instance`

L'exemple de code suivant montre comment utiliser`Remove-EC2Instance`.

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 API plus de détails, consultez la section TerminateInstancesRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EC2InternetGateway`

L'exemple de code suivant montre comment utiliser`Remove-EC2InternetGateway`.

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 API plus de détails, consultez la section DeleteInternetGatewayRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EC2KeyPair`

L'exemple de code suivant montre comment utiliser`Remove-EC2KeyPair`.

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 API plus de détails, consultez la section DeleteKeyPairRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EC2NetworkAcl`

L'exemple de code suivant montre comment utiliser`Remove-EC2NetworkAcl`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le réseau ACL spécifié. 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 API plus de détails, consultez la section DeleteNetworkAclRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EC2NetworkAclEntry`

L'exemple de code suivant montre comment utiliser`Remove-EC2NetworkAclEntry`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la règle spécifiée du réseau spécifiéACL. 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 API plus de détails, consultez la section DeleteNetworkAclEntryRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EC2NetworkInterface`

L'exemple de code suivant montre comment utiliser`Remove-EC2NetworkInterface`.

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 API plus de détails, consultez la section DeleteNetworkInterfaceRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EC2PlacementGroup`

L'exemple de code suivant montre comment utiliser`Remove-EC2PlacementGroup`.

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 API plus de détails, consultez la section DeletePlacementGroupRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EC2Route`

L'exemple de code suivant montre comment utiliser`Remove-EC2Route`.

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 API plus de détails, consultez la section DeleteRouteRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EC2RouteTable`

L'exemple de code suivant montre comment utiliser`Remove-EC2RouteTable`.

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 API plus de détails, consultez la section DeleteRouteTableRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EC2SecurityGroup`

L'exemple de code suivant montre comment utiliser`Remove-EC2SecurityGroup`.

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 -Classic.

```
Remove-EC2SecurityGroup -GroupName my-security-group -Force
```

- Pour API plus de détails, consultez la section DeleteSecurityGroupRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EC2Snapshot`

L'exemple de code suivant montre comment utiliser`Remove-EC2Snapshot`.

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 API plus de détails, consultez la section DeleteSnapshotRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EC2SpotDatafeedSubscription`

L'exemple de code suivant montre comment utiliser`Remove-EC2SpotDatafeedSubscription`.

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 API plus de détails, consultez la section DeleteSpotDatafeedSubscriptionRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EC2Subnet`

L'exemple de code suivant montre comment utiliser`Remove-EC2Subnet`.

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 API plus de détails, consultez la section DeleteSubnetRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EC2Tag`

L'exemple de code suivant montre comment utiliser`Remove-EC2Tag`.

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 API plus de détails, consultez la section DeleteTagsRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EC2Volume`

L'exemple de code suivant montre comment utiliser`Remove-EC2Volume`.

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 API plus de détails, consultez la section DeleteVolumeRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EC2Vpc`

L'exemple de code suivant montre comment utiliser`Remove-EC2Vpc`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le paramètre spécifiéVPC. 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 API plus de détails, consultez la section DeleteVpcRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EC2VpnConnection`

L'exemple de code suivant montre comment utiliser`Remove-EC2VpnConnection`.

Outils pour PowerShell

Exemple 1 : cet exemple supprime la VPN connexion 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 API plus de détails, consultez la section DeleteVpnConnectionRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EC2VpnConnectionRoute`

L'exemple de code suivant montre comment utiliser`Remove-EC2VpnConnectionRoute`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la route statique spécifiée de la VPN connexion 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 API plus de détails, consultez la section DeleteVpnConnectionRouteRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EC2VpnGateway`

L'exemple de code suivant montre comment utiliser`Remove-EC2VpnGateway`.

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 API plus de détails, consultez la section DeleteVpnGatewayRéférence des AWS Tools for PowerShell applets de commande.

### `Request-EC2SpotFleet`

L'exemple de code suivant montre comment utiliser`Request-EC2SpotFleet`.

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 est VPC uniquement compatibleEC2, le parc Spot lance les instances dans la zone de disponibilité la moins chère dotée d'un sous-réseau par défaut. Si votre compte est compatible avec EC2 -Classic, le parc Spot lance les instances dans EC2 -Classic 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 API plus de détails, consultez la section RequestSpotFleetRéférence des AWS Tools for PowerShell applets de commande.

### `Request-EC2SpotInstance`

L'exemple de code suivant montre comment utiliser`Request-EC2SpotInstance`.

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 sous-réseau 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 API plus de détails, consultez la section RequestSpotInstancesRéférence des AWS Tools for PowerShell applets de commande.

### `Reset-EC2ImageAttribute`

L'exemple de code suivant montre comment utiliser`Reset-EC2ImageAttribute`.

Outils pour PowerShell

Exemple 1 : Cet exemple rétablit la valeur par défaut de l'attribut launchPermission « ». Par défaut, AMIs sont privés.

```
Reset-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

- Pour API plus de détails, consultez la section ResetImageAttributeRéférence des AWS Tools for PowerShell applets de commande.

### `Reset-EC2InstanceAttribute`

L'exemple de code suivant montre comment utiliser`Reset-EC2InstanceAttribute`.

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 API plus de détails, consultez la section ResetInstanceAttributeRéférence des AWS Tools for PowerShell applets de commande.

### `Reset-EC2NetworkInterfaceAttribute`

L'exemple de code suivant montre comment utiliser`Reset-EC2NetworkInterfaceAttribute`.

Outils pour PowerShell

Exemple 1 : Cet exemple réinitialise la vérification source/destination pour l'interface réseau spécifiée.

```
Reset-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -SourceDestCheck
```

- Pour API plus de détails, consultez la section ResetNetworkInterfaceAttributeRéférence des AWS Tools for PowerShell applets de commande.

### `Reset-EC2SnapshotAttribute`

L'exemple de code suivant montre comment utiliser`Reset-EC2SnapshotAttribute`.

Outils pour PowerShell

Exemple 1 : Cet exemple réinitialise l'attribut spécifié pour le cliché spécifié.

```
Reset-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission
```

- Pour API plus de détails, consultez la section ResetSnapshotAttributeRéférence des AWS Tools for PowerShell applets de commande.

### `Restart-EC2Instance`

L'exemple de code suivant montre comment utiliser`Restart-EC2Instance`.

Outils pour PowerShell

Exemple 1 : Cet exemple redémarre l'instance spécifiée.

```
Restart-EC2Instance -InstanceId i-12345678
```

- Pour API plus de détails, consultez la section RebootInstancesRéférence des AWS Tools for PowerShell applets de commande.

### `Revoke-EC2SecurityGroupEgress`

L'exemple de code suivant montre comment utiliser`Revoke-EC2SecurityGroupEgress`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la règle du groupe de sécurité spécifié pour EC2 -VPC. Cela révoque l'accès à la plage d'adresses IP spécifiée sur le TCP port 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 TCP port 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 API plus de détails, consultez la section RevokeSecurityGroupEgressRéférence des AWS Tools for PowerShell applets de commande.

### `Revoke-EC2SecurityGroupIngress`

L'exemple de code suivant montre comment utiliser`Revoke-EC2SecurityGroupIngress`.

Exemple 1 : Cet exemple révoque l'accès au TCP port 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 en utilisant l'ID du groupe de sécurité et non le 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 TCP port 22 à partir de la plage d'adresses spécifiée pour le groupe de sécurité spécifié pour EC2 -Classic. 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 API plus de détails, consultez la section RevokeSecurityGroupIngressRéférence des AWS Tools for PowerShell applets de commande.

### `Send-EC2InstanceStatus`

L'exemple de code suivant montre comment utiliser`Send-EC2InstanceStatus`.

Outils pour PowerShell

Exemple 1 : Cet exemple indique le statut de l'instance spécifiée.

```
Send-EC2InstanceStatus -Instance i-12345678 -Status impaired -ReasonCode
 unresponsive
```

- Pour API plus de détails, consultez la section ReportInstanceStatusRéférence des AWS Tools for PowerShell applets de commande.

### `Set-EC2NetworkAclAssociation`

L'exemple de code suivant montre comment utiliser`Set-EC2NetworkAclAssociation`.

Outils pour PowerShell

Exemple 1 : Cet exemple associe le réseau ACL spécifié au sous-réseau correspondant à l'ACLassociation réseau spécifiée.

```
Set-EC2NetworkAclAssociation -NetworkAclId acl-12345678 -AssociationId
 aclassoc-1a2b3c4d
```

Sortie :

```
aclassoc-87654321
```

- Pour API plus de détails, consultez la section ReplaceNetworkAclAssociationRéférence des AWS Tools for PowerShell applets de commande.

### `Set-EC2NetworkAclEntry`

L'exemple de code suivant montre comment utiliser`Set-EC2NetworkAclEntry`.

Outils pour PowerShell

Exemple 1 : Cet exemple remplace l'entrée spécifiée pour le réseau spécifiéACL. 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 API plus de détails, consultez la section ReplaceNetworkAclEntryRéférence des AWS Tools for PowerShell applets de commande.

### `Set-EC2Route`

L'exemple de code suivant montre comment utiliser`Set-EC2Route`.

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 API plus de détails, consultez la section ReplaceRouteRéférence des AWS Tools for PowerShell applets de commande.

### `Set-EC2RouteTableAssociation`

L'exemple de code suivant montre comment utiliser`Set-EC2RouteTableAssociation`.

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 API plus de détails, consultez la section ReplaceRouteTableAssociationRéférence des AWS Tools for PowerShell applets de commande.

### `Start-EC2Instance`

L'exemple de code suivant montre comment utiliser`Start-EC2Instance`.

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 versStart-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 API plus de détails, consultez la section StartInstancesRéférence des AWS Tools for PowerShell applets de commande.

### `Start-EC2InstanceMonitoring`

L'exemple de code suivant montre comment utiliser`Start-EC2InstanceMonitoring`.

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 API plus de détails, consultez la section MonitorInstancesRéférence des AWS Tools for PowerShell applets de commande.

### `Stop-EC2ImportTask`

L'exemple de code suivant montre comment utiliser`Stop-EC2ImportTask`.

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 API plus de détails, consultez la section CancelImportTaskRéférence des AWS Tools for PowerShell applets de commande.

### `Stop-EC2Instance`

L'exemple de code suivant montre comment utiliser`Stop-EC2Instance`.

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 API plus de détails, consultez la section StopInstancesRéférence des AWS Tools for PowerShell applets de commande.

### `Stop-EC2InstanceMonitoring`

L'exemple de code suivant montre comment utiliser`Stop-EC2InstanceMonitoring`.

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 API plus de détails, consultez la section UnmonitorInstancesRéférence des AWS Tools for PowerShell applets de commande.

### `Stop-EC2SpotFleetRequest`

L'exemple de code suivant montre comment utiliser`Stop-EC2SpotFleetRequest`.

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 API plus de détails, consultez la section CancelSpotFleetRequestsRéférence des AWS Tools for PowerShell applets de commande.

### `Stop-EC2SpotInstanceRequest`

L'exemple de code suivant montre comment utiliser`Stop-EC2SpotInstanceRequest`.

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 API plus de détails, consultez la section CancelSpotInstanceRequestsRéférence des AWS Tools for PowerShell applets de commande.

### `Unregister-EC2Address`

L'exemple de code suivant montre comment utiliser`Unregister-EC2Address`.

Outils pour PowerShell

Exemple 1 : Cet exemple dissocie l'adresse IP élastique spécifiée de l'instance spécifiée dans unVPC.

```
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 API plus de détails, consultez la section DisassociateAddressRéférence des AWS Tools for PowerShell applets de commande.

### `Unregister-EC2Image`

L'exemple de code suivant montre comment utiliser`Unregister-EC2Image`.

Outils pour PowerShell

Exemple 1 : Cet exemple annule l'enregistrement du fichier spécifié. AMI

```
Unregister-EC2Image -ImageId ami-12345678
```

- Pour API plus de détails, consultez la section DeregisterImageRéférence des AWS Tools for PowerShell applets de commande.

### `Unregister-EC2PrivateIpAddress`

L'exemple de code suivant montre comment utiliser`Unregister-EC2PrivateIpAddress`.

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 API plus de détails, consultez la section UnassignPrivateIpAddressesRéférence des AWS Tools for PowerShell applets de commande.

### `Unregister-EC2RouteTable`

L'exemple de code suivant montre comment utiliser`Unregister-EC2RouteTable`.

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 API plus de détails, consultez la section DisassociateRouteTableRéférence des AWS Tools for PowerShell applets de commande.

### `Update-EC2SecurityGroupRuleIngressDescription`

L'exemple de code suivant montre comment utiliser`Update-EC2SecurityGroupRuleIngressDescription`.

Outils pour PowerShell

Exemple 1 : met à jour la description d'une règle de groupe de sécurité d'entrée (entrante) existante.

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupRuleId
 "sgr-1234567890"
$ruleWithUpdatedDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
  "SecurityGroupRuleId" = $existingInboundRule.SecurityGroupRuleId
  "Description" = "Updated rule description"
}

Update-EC2SecurityGroupRuleIngressDescription -GroupId $existingInboundRule.GroupId
 -SecurityGroupRuleDescription $ruleWithUpdatedDescription
```

Exemple 2 : Supprime la description d'une règle de groupe de sécurité d'entrée (entrante) existante (en omettant le paramètre dans la demande).

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupRuleId
 "sgr-1234567890"
$ruleWithoutDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
  "SecurityGroupRuleId" = $existingInboundRule.SecurityGroupRuleId
}

Update-EC2SecurityGroupRuleIngressDescription -GroupId $existingInboundRule.GroupId
 -SecurityGroupRuleDescription $ruleWithoutDescription
```

- Pour API plus de détails, consultez la section UpdateSecurityGroupRuleDescriptionsIngressRéférence des AWS Tools for PowerShell applets de commande.

# ECRExemples Amazon utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS Tools for PowerShell aide d'AmazonECR.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `Get-ECRLoginCommand`

L'exemple de code suivant montre comment utiliser`Get-ECRLoginCommand`.

Outils pour PowerShell

Exemple 1 : renvoie un fichier PSObject contenant des informations de connexion qui peuvent être utilisées pour s'authentifier auprès de n'importe quel ECR registre Amazon auquel votre IAM

principal a accès. Les informations d'identification et le point de terminaison régional requis pour l'appel afin d'obtenir le jeton d'autorisation sont obtenus à partir des paramètres par défaut du shell (définis par les **`Initialize-AWSDefaultConfiguration`** applets de commande **`Set-AWSCredential/Set-DefaultAWSRegion`** or). Vous pouvez utiliser la propriété Command avec Invoke-Expression pour vous connecter au registre spécifié ou utiliser les informations d'identification renvoyées dans d'autres outils nécessitant une connexion.

```
Get-ECRLoginCommand
```

Sortie :

```
Username      : AWS
Password      : eyJwYXlsb2Fk...kRBVEFfS0VZIn0=
ProxyEndpoint : https://123456789012.dkr.ecr.us-west-2.amazonaws.com
Endpoint      : https://123456789012.dkr.ecr.us-west-2.amazonaws.com
ExpiresAt     : 9/26/2017 6:08:23 AM
Command       : docker login --username AWS --password
 eyJwYXlsb2Fk...kRBVEFfS0VZIn0= https://123456789012.dkr.ecr.us-west-2.amazonaws.com
```

Exemple 2 : récupère un PSObject conteneur d'informations de connexion que vous utilisez comme entrée pour une commande de connexion docker. Vous pouvez spécifier n'importe quel ECR registre Amazon auprès URI duquel vous souhaitez vous authentifier, à condition que votre IAM principal ait accès à ce registre.

```
(Get-ECRLoginCommand).Password | docker login --username AWS --password-stdin
 012345678910.dkr.ecr.us-east-1.amazonaws.com
```

- Pour API plus de détails, consultez la section Référence des AWS Tools for PowerShell applets de commande Get- ECRLoginCommand in.

# ECSExemples Amazon utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS Tools for PowerShell aide d'AmazonECS.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `Get-ECSClusterDetail`

L'exemple de code suivant montre comment utiliser`Get-ECSClusterDetail`.

Outils pour PowerShell

Exemple 1 : cette applet de commande décrit un ou plusieurs de vos ECS clusters.

```
Get-ECSClusterDetail -Cluster "LAB-ECS-CL" -Include SETTINGS | Select-Object *
```

Sortie :

```
LoggedAt         : 12/27/2019 9:27:41 PM
Clusters         : {LAB-ECS-CL}
Failures         : {}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 396
HttpStatusCode   : OK
```

- Pour API plus de détails, consultez la section DescribeClustersRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ECSClusterList`

L'exemple de code suivant montre comment utiliser`Get-ECSClusterList`.

Outils pour PowerShell

Exemple 1 : cette applet de commande renvoie une liste des clusters existantsECS.

```
Get-ECSClusterList
```

Sortie :

```
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS-CL
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS
```

- Pour API plus de détails, consultez la section ListClustersRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ECSClusterService`

L'exemple de code suivant montre comment utiliser`Get-ECSClusterService`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie tous les services exécutés dans votre cluster par défaut.

```
Get-ECSClusterService
```

Exemple 2 : Cet exemple répertorie tous les services exécutés dans le cluster spécifié.

```
Get-ECSClusterService -Cluster myCluster
```

Exemple 3 : Cet exemple répertorie les services exécutés dans le cluster spécifié, en récupérant un maximum de 10 détails de service à la fois.

```
$nextToken = $null
do
{
    Get-ECSClusterService -Cluster myCluster -MaxResult 10 -NextToken $nextToken
    $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Pour API plus de détails, consultez la section ListServicesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ECSService`

L'exemple de code suivant montre comment utiliser`Get-ECSService`.

Outils pour PowerShell

Exemple 1 : Cet exemple montre comment récupérer les détails d'un service spécifique à partir de votre cluster par défaut.

```
Get-ECSService -Service my-hhtp-service
```

Exemple 2 : Cet exemple montre comment récupérer les détails d'un service spécifique exécuté dans le cluster nommé.

```
Get-ECSService -Cluster myCluster -Service my-hhtp-service
```

- Pour API plus de détails, consultez la section DescribeServicesRéférence des AWS Tools for PowerShell applets de commande.

### `New-ECSCluster`

L'exemple de code suivant montre comment utiliser`New-ECSCluster`.

Outils pour PowerShell

Exemple 1 : Cette applet de commande crée un nouveau cluster AmazonECS.

```
New-ECSCluster -ClusterName "LAB-ECS-CL" -Setting @{Name="containerInsights";
 Value="enabled"}
```

Sortie :

```
ActiveServicesCount              : 0
Attachments                      : {}
AttachmentsStatus                :
CapacityProviders                : {}
ClusterArn                       : arn:aws:ecs:us-west-2:012345678912:cluster/LAB-
ECS-CL
ClusterName                      : LAB-ECS-CL
DefaultCapacityProviderStrategy  : {}
PendingTasksCount                : 0
RegisteredContainerInstancesCount : 0
RunningTasksCount                : 0
```

```
Settings                          : {containerInsights}
Statistics                        : {}
Status                            : ACTIVE
Tags                              : {}
```

- Pour API plus de détails, consultez la section CreateClusterRéférence des AWS Tools for PowerShell applets de commande.

### `New-ECSService`

L'exemple de code suivant montre comment utiliser`New-ECSService`.

Outils pour PowerShell

Exemple 1 : Cet exemple de commande crée un service dans votre cluster par défaut appelé `ecs-simple-service`. Le service utilise la définition de tâche `ecs-demo` et il gère 10 instanciations de cette tâche.

```
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10
```

Exemple 2 : Cet exemple de commande crée un service derrière un équilibreur de charge dans votre cluster par défaut appelé `ecs-simple-service`. Le service utilise la définition de tâche `ecs-demo` et il gère 10 instanciations de cette tâche.

```
$lb = @{
    LoadBalancerName = "EC2Contai-EcsElast-S06278JGSJCM"
    ContainerName = "simple-demo"
    ContainerPort = 80
}
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10 -LoadBalancer $lb
```

- Pour API plus de détails, consultez la section CreateServiceRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-ECSCluster`

L'exemple de code suivant montre comment utiliser`Remove-ECSCluster`.

Outils pour PowerShell

Exemple 1 : cette applet de commande supprime le cluster spécifié. ECS Vous devez désenregistrer toutes les instances de conteneur de ce cluster avant de pouvoir le supprimer.

```
Remove-ECSCluster -Cluster "LAB-ECS"
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ECSCluster (DeleteCluster)" on target "LAB-ECS".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Pour API plus de détails, consultez la section DeleteClusterRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-ECSService`

L'exemple de code suivant montre comment utiliser`Remove-ECSService`.

Outils pour PowerShell

Exemple 1 : Supprime le service nommé « my-http-service » dans le cluster par défaut. Le service doit avoir le nombre souhaité et le nombre de points courants de 0 pour que vous puissiez le supprimer. Vous êtes invité à confirmer avant que la commande ne soit exécutée. Pour contourner l'invite de confirmation, ajoutez le commutateur -Force.

```
Remove-ECSService -Service my-http-service
```

Exemple 2 : Supprime le service nommé « my-http-service » dans le cluster nommé.

```
Remove-ECSService -Cluster myCluster -Service my-http-service
```

- Pour API plus de détails, consultez la section DeleteServiceRéférence des AWS Tools for PowerShell applets de commande.

### `Update-ECSClusterSetting`

L'exemple de code suivant montre comment utiliser`Update-ECSClusterSetting`.

Outils pour PowerShell

Exemple 1 : cette applet de commande modifie les paramètres à utiliser pour un cluster. ECS

```
Update-ECSClusterSetting -Cluster "LAB-ECS-CL" -Setting @{Name="containerInsights";
 Value="disabled"}
```

Sortie :

```
ActiveServicesCount               : 0
Attachments                       : {}
AttachmentsStatus                 :
CapacityProviders                 : {}
ClusterArn                        : arn:aws:ecs:us-west-2:012345678912:cluster/LAB-
ECS-CL
ClusterName                       : LAB-ECS-CL
DefaultCapacityProviderStrategy   : {}
PendingTasksCount                 : 0
RegisteredContainerInstancesCount : 0
RunningTasksCount                 : 0
Settings                          : {containerInsights}
Statistics                        : {}
Status                            : ACTIVE
Tags                              : {}
```

- Pour API plus de détails, consultez la section UpdateClusterSettingsRéférence des AWS Tools for PowerShell applets de commande.

### `Update-ECSService`

L'exemple de code suivant montre comment utiliser`Update-ECSService`.

Outils pour PowerShell

Exemple 1 : Cet exemple de commande met à jour le service my-http-service `` pour utiliser la définition de tâche amazon-ecs-sample ``.

```
Update-ECSService -Service my-http-service -TaskDefinition amazon-ecs-sample
```

Exemple 2 : Cet exemple de commande met à jour le nombre souhaité du service my-http-service `` à 10.

```
Update-ECSService -Service my-http-service -DesiredCount 10
```

- Pour API plus de détails, consultez la section UpdateServiceRéférence des AWS Tools for PowerShell applets de commande.

# EFSExemples Amazon utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS Tools for PowerShell aide d'AmazonEFS.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `Edit-EFSMountTargetSecurityGroup`

L'exemple de code suivant montre comment utiliser`Edit-EFSMountTargetSecurityGroup`.

Outils pour PowerShell

Exemple 1 : met à jour les groupes de sécurité en vigueur pour la cible de montage spécifiée. Jusqu'à 5 peuvent être spécifiés, au format « sg-xxxxxxxx ».

```
Edit-EFSMountTargetSecurityGroup -MountTargetId fsmt-1a2b3c4d -SecurityGroup sg-
group1,sg-group3
```

- Pour API plus de détails, consultez la section ModifyMountTargetSecurityGroupsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EFSFileSystem`

L'exemple de code suivant montre comment utiliser`Get-EFSFileSystem`.

Outils pour PowerShell

Exemple 1 : renvoie la collection de tous les systèmes de fichiers appartenant au compte de l'appelant dans la région.

```
Get-EFSFileSystem
```

Sortie :

```
CreationTime         : 5/26/2015 4:02:38 PM
CreationToken        : 1a2bff54-85e0-4747-bd95-7bc172c4f555
FileSystemId         : fs-1a2b3c4d
LifeCycleState       : available
Name                 :
NumberOfMountTargets : 0
OwnerId              : 123456789012
SizeInBytes          : Amazon.ElasticFileSystem.Model.FileSystemSize

CreationTime         : 5/26/2015 4:06:23 PM
CreationToken        : 2b4daa14-85e0-4747-bd95-7bc172c4f555
FileSystemId         : fs-4d3c2b1a
...
```

Exemple 2 : renvoie les détails du système de fichiers spécifié.

```
Get-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

Exemple 3 : renvoie les détails d'un système de fichiers à l'aide du jeton de création d'idempotencie spécifié au moment de la création du système de fichiers.

```
Get-EFSFileSystem -CreationToken 1a2bff54-85e0-4747-bd95-7bc172c4f555
```

- Pour API plus de détails, consultez la section DescribeFileSystemsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EFSMountTarget`

L'exemple de code suivant montre comment utiliser`Get-EFSMountTarget`.

Outils pour PowerShell

Exemple 1 : renvoie la collection de cibles de montage associées au système de fichiers spécifié.

```
Get-EFSMountTarget -FileSystemId fs-1a2b3c4d
```

Sortie :

```
FileSystemId       : fs-1a2b3c4d
IpAddress          : 10.0.0.131
LifeCycleState     : available
MountTargetId      : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId            : 123456789012
SubnetId           : subnet-1a2b3c4d
```

- Pour API plus de détails, consultez la section DescribeMountTargetsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EFSMountTargetSecurityGroup`

L'exemple de code suivant montre comment utiliser`Get-EFSMountTargetSecurityGroup`.

Outils pour PowerShell

Exemple 1 : renvoie les identifiants des groupes de sécurité actuellement affectés à l'interface réseau associée à la cible de montage.

```
Get-EFSMountTargetSecurityGroup -MountTargetId fsmt-1a2b3c4d
```

Sortie :

```
sg-1a2b3c4d
```

- Pour API plus de détails, consultez la section DescribeMountTargetSecurityGroupsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EFSTag`

L'exemple de code suivant montre comment utiliser`Get-EFSTag`.

Outils pour PowerShell

Exemple 1 : renvoie la collection de balises actuellement associées au système de fichiers spécifié.

```
Get-EFSTag -FileSystemId fs-1a2b3c4d
```

Sortie :

```
Key         Value
---         -----
Name        My File System
tagkey1     tagvalue1
tagkey2     tagvalue2
```

- Pour API plus de détails, consultez la section DescribeTagsRéférence des AWS Tools for PowerShell applets de commande.

### `New-EFSFileSystem`

L'exemple de code suivant montre comment utiliser`New-EFSFileSystem`.

Outils pour PowerShell

Exemple 1 : crée un nouveau système de fichiers vide. Le jeton utilisé pour garantir la création idempotente sera généré automatiquement et sera accessible depuis le **`CreationToken`** membre de l'objet renvoyé.

```
New-EFSFileSystem
```

Sortie :

```
CreationTime         : 5/26/2015 4:02:38 PM
CreationToken        : 1a2bff54-85e0-4747-bd95-7bc172c4f555
FileSystemId         : fs-1a2b3c4d
LifeCycleState       : creating
Name                 :
```

```
NumberOfMountTargets : 0
OwnerId              : 123456789012
SizeInBytes          : Amazon.ElasticFileSystem.Model.FileSystemSize
```

Exemple 2 : crée un nouveau système de fichiers vide à l'aide d'un jeton personnalisé pour garantir une création idempotente.

```
New-EFSFileSystem -CreationToken "MyUniqueToken"
```

- Pour API plus de détails, consultez la section CreateFileSystemRéférence des AWS Tools for PowerShell applets de commande.

### `New-EFSMountTarget`

L'exemple de code suivant montre comment utiliser`New-EFSMountTarget`.

Outils pour PowerShell

Exemple 1 : crée une nouvelle cible de montage pour un système de fichiers. Le sous-réseau spécifié sera utilisé pour déterminer le cloud privé virtuel (VPC) dans lequel la cible de montage sera créée et l'adresse IP qui sera attribuée automatiquement (à partir de la plage d'adresses du sous-réseau). L'adresse IP attribuée peut ensuite être utilisée pour monter ce système de fichiers sur une EC2 instance Amazon. Aucun groupe de sécurité n'ayant été spécifié, l'interface réseau créée pour la cible est associée au groupe de sécurité par défaut pour le sous-réseau. VPC

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d
```

Sortie :

```
FileSystemId       : fs-1a2b3c4d
IpAddress          : 10.0.0.131
LifeCycleState     : creating
MountTargetId      : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId            : 123456789012
SubnetId           : subnet-1a2b3c4d
```

Exemple 2 : crée une nouvelle cible de montage pour le système de fichiers spécifié avec une adresse IP attribuée automatiquement. L'interface réseau créée pour la cible de montage est

associée aux groupes de sécurité spécifiés (jusqu'à 5, au format « sg-xxxxxxxx », peuvent être spécifiés).

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -
SecurityGroup sg-group1,sg-group2,sg-group3
```

Exemple 3 : crée une nouvelle cible de montage pour le système de fichiers spécifié avec l'adresse IP spécifiée.

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -IpAddress
 10.0.0.131
```

- Pour API plus de détails, consultez la section CreateMountTargetRéférence des AWS Tools for PowerShell applets de commande.

### `New-EFSTag`

L'exemple de code suivant montre comment utiliser`New-EFSTag`.

Outils pour PowerShell

Exemple 1 : applique la collection de balises au système de fichiers spécifié. Si une balise dont la clé est spécifiée existe déjà dans le système de fichiers, la valeur de la balise est mise à jour.

```
New-EFSTag -FileSystemId fs-1a2b3c4d -Tag
 @{Key="tagkey1";Value="tagvalue1"},@{Key="tagkey2";Value="tagvalue2"}
```

Exemple 2 : définit la balise de nom pour le système de fichiers spécifié. Cette valeur est renvoyée avec d'autres détails du système de fichiers lorsque l'EFSFileSystemapplet de commande Get- est utilisée.

```
New-EFSTag -FileSystemId fs-1a2b3c4d -Tag @{Key="Name";Value="My File System"}
```

- Pour API plus de détails, consultez la section CreateTagsRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EFSFileSystem`

L'exemple de code suivant montre comment utiliser`Remove-EFSFileSystem`.

Outils pour PowerShell

Exemple 1 : Supprime le système de fichiers spécifié qui n'est plus utilisé (si le système de fichiers possède des cibles de montage, elles doivent d'abord être supprimées). Vous êtes invité à confirmer avant que l'applet de commande ne démarre. Pour supprimer la confirmation, utilisez le **`-Force`** commutateur.

```
Remove-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

- Pour API plus de détails, consultez la section DeleteFileSystemRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EFSMountTarget`

L'exemple de code suivant montre comment utiliser`Remove-EFSMountTarget`.

Outils pour PowerShell

Exemple 1 : Supprime la cible de montage spécifiée. Vous êtes invité à confirmer avant que l'opération ne se poursuive. Pour supprimer l'invite, utilisez le **`-Force`** commutateur. Notez que cette opération interrompt de force tous les montages du système de fichiers via la cible. Vous pouvez envisager de démonter le système de fichiers avant d'exécuter cette commande, si possible.

```
Remove-EFSMountTarget -MountTargetId fsmt-1a2b3c4d
```

- Pour API plus de détails, consultez la section DeleteMountTargetRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EFSTag`

L'exemple de code suivant montre comment utiliser`Remove-EFSTag`.

Outils pour PowerShell

Exemple 1 : Supprime la collection d'une ou de plusieurs balises d'un système de fichiers. Vous êtes invité à confirmer avant que l'applet de commande ne démarre. Pour supprimer la confirmation, utilisez le **`-Force`** commutateur.

```
Remove-EFSTag -FileSystemId fs-1a2b3c4d -TagKey "tagkey1","tagkey2"
```

- Pour API plus de détails, consultez la section DeleteTagsRéférence des AWS Tools for PowerShell applets de commande.

# EKSExemples Amazon utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS Tools for PowerShell aide d'AmazonEKS.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `Add-EKSResourceTag`

L'exemple de code suivant montre comment utiliser`Add-EKSResourceTag`.

Outils pour PowerShell

Exemple 1 : Cette applet de commande associe les balises spécifiées à une ressource avec les balises spécifiées. resourceArn

```
Add-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD" -
Tag @{Name = "EKSPRODCLUSTER"}
```

- Pour API plus de détails, consultez la section TagResourceRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EKSCluster`

L'exemple de code suivant montre comment utiliser`Get-EKSCluster`.

Outils pour PowerShell

Exemple 1 : Cette applet de commande renvoie des informations descriptives sur un cluster AmazonEKS.

```
Get-EKSCluster -Name "PROD"
```

Sortie :

```
Arn                  : arn:aws:eks:us-west-2:012345678912:cluster/PROD
CertificateAuthority : Amazon.EKS.Model.Certificate
ClientRequestToken   :
CreatedAt            : 12/25/2019 6:46:17 AM
Endpoint             : https://669608765450FBBE54D1D78A3D71B72C.gr8.us-
west-2.eks.amazonaws.com
Identity             : Amazon.EKS.Model.Identity
Logging              : Amazon.EKS.Model.Logging
Name                 : PROD
PlatformVersion      : eks.7
ResourcesVpcConfig   : Amazon.EKS.Model.VpcConfigResponse
RoleArn              : arn:aws:iam::012345678912:role/eks-iam-role
Status               : ACTIVE
Tags                 : {}
Version              : 1.14
```

- Pour API plus de détails, consultez la section DescribeClusterRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EKSClusterList`

L'exemple de code suivant montre comment utiliser`Get-EKSClusterList`.

Outils pour PowerShell

Exemple 1 : Cette applet de commande répertorie les EKS clusters Amazon présents Compte AWS dans la région spécifiée.

```
Get-EKSClusterList
```

Sortie :

```
 PROD
```

- Pour API plus de détails, consultez la section ListClustersRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EKSFargateProfile`

L'exemple de code suivant montre comment utiliser`Get-EKSFargateProfile`.

Outils pour PowerShell

Exemple 1 : Cette applet de commande renvoie des informations descriptives sur un profil Fargate AWS .

```
Get-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

Sortie :

```
ClusterName         : TEST
CreatedAt           : 12/26/2019 12:34:47 PM
FargateProfileArn   : arn:aws:eks:us-east-2:012345678912:fargateprofile/TEST/
EKSFargate/42b7a119-e16b-a279-ce97-bdf303adec92
FargateProfileName  : EKSFargate
PodExecutionRoleArn : arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole
Selectors           : {Amazon.EKS.Model.FargateProfileSelector}
Status              : ACTIVE
Subnets             : {subnet-0cd976f08d5fbfaae, subnet-02f6ff500ff2067a0}
Tags                : {}
```

- Pour API plus de détails, consultez la section DescribeFargateProfileRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EKSFargateProfileList`

L'exemple de code suivant montre comment utiliser`Get-EKSFargateProfileList`.

Outils pour PowerShell

Exemple 1 : Cette applet de commande répertorie les profils AWS Fargate associés au cluster spécifié dans votre région spécifiée. Compte AWS

```
Get-EKSFargateProfileList -ClusterName "TEST"
```

Sortie :

```
EKSFargate
EKSFargateProfile
```

- Pour API plus de détails, consultez la section ListFargateProfilesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EKSNodegroup`

L'exemple de code suivant montre comment utiliser`Get-EKSNodegroup`.

Outils pour PowerShell

Exemple 1 : Cette applet de commande renvoie des informations descriptives sur un groupe de EKS nœuds Amazon.

```
Get-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

Sortie :

```
AmiType        : AL2_x86_64
ClusterName    : PROD
CreatedAt      : 12/25/2019 10:16:45 AM
DiskSize       : 40
Health         : Amazon.EKS.Model.NodegroupHealth
InstanceTypes  : {t3.large}
Labels         : {}
ModifiedAt     : 12/25/2019 10:16:45 AM
NodegroupArn   : arn:aws:eks:us-west-2:012345678912:nodegroup/PROD/
ProdEKSNodeGroup/7eb79e47-82b6-04d9-e984-95110db6fa85
NodegroupName  : ProdEKSNodeGroup
NodeRole       : arn:aws:iam::012345678912:role/NodeInstanceRole
ReleaseVersion : 1.14.7-20190927
```

```
RemoteAccess   :
Resources      :
ScalingConfig  : Amazon.EKS.Model.NodegroupScalingConfig
Status         : CREATING
Subnets        : {subnet-0d1a9fff35efa7691, subnet-0a3f4928edbc224d4}
Tags           : {}
Version        : 1.14
```

- Pour API plus de détails, consultez la section DescribeNodegroupRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EKSNodegroupList`

L'exemple de code suivant montre comment utiliser`Get-EKSNodegroupList`.

Outils pour PowerShell

Exemple 1 : Cette applet de commande répertorie les groupes de EKS nœuds Amazon associés au cluster spécifié Compte AWS dans votre région.

```
Get-EKSNodegroupList -ClusterName PROD
```

Sortie :

```
ProdEKSNodeGroup
```

- Pour API plus de détails, consultez la section ListNodegroupsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EKSResourceTag`

L'exemple de code suivant montre comment utiliser`Get-EKSResourceTag`.

Outils pour PowerShell

Exemple 1 : Cette applet de commande répertorie les balises d'une ressource AmazonEKS.

```
Get-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
```

Sortie :

```
Key  Value
---  -----
Name EKSPRODCLUSTER
```

- Pour API plus de détails, consultez la section ListTagsForResourceRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EKSUpdate`

L'exemple de code suivant montre comment utiliser`Get-EKSUpdate`.

Outils pour PowerShell

Exemple 1 : Cette applet de commande renvoie des informations descriptives concernant une mise à jour concernant votre EKS cluster Amazon ou le groupe de nœuds gérés associé.

```
Get-EKSUpdate -Name "PROD" -UpdateId "ee708232-7d2e-4ed7-9270-d0b5176f0726"
```

Sortie :

```
CreatedAt : 12/25/2019 5:03:07 PM
Errors    : {}
Id        : ee708232-7d2e-4ed7-9270-d0b5176f0726
Params    : {Amazon.EKS.Model.UpdateParam}
Status    : Successful
Type      : LoggingUpdate
```

- Pour API plus de détails, consultez la section DescribeUpdateRéférence des AWS Tools for PowerShell applets de commande.

### `Get-EKSUpdateList`

L'exemple de code suivant montre comment utiliser`Get-EKSUpdateList`.

Outils pour PowerShell

Exemple 1 : Cette applet de commande répertorie les mises à jour associées à un EKS cluster Amazon ou à un groupe de nœuds gérés dans votre Compte AWS région spécifiée.

```
Get-EKSUpdateList -Name "PROD"
```

Sortie :

```
ee708232-7d2e-4ed7-9270-d0b5176f0726
```

- Pour API plus de détails, consultez la section ListUpdatesRéférence des AWS Tools for PowerShell applets de commande.

### `New-EKSCluster`

L'exemple de code suivant montre comment utiliser`New-EKSCluster`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un nouveau cluster appelé « prod ».

```
New-EKSCluster -Name prod -ResourcesVpcConfig
 @{SubnetIds=@("subnet-0a1b2c3d","subnet-3a2b1c0d");SecurityGroupIds="sg-6979fe18"}
 -RoleArn "arn:aws:iam::012345678901:role/eks-service-role"
```

Sortie :

```
Arn                 : arn:aws:eks:us-west-2:012345678901:cluster/prod
CertificateAuthority : Amazon.EKS.Model.Certificate
ClientRequestToken  :
CreatedAt           : 12/10/2018 9:25:31 PM
Endpoint            :
Name                : prod
PlatformVersion     : eks.3
ResourcesVpcConfig  : Amazon.EKS.Model.VpcConfigResponse
RoleArn             : arn:aws:iam::012345678901:role/eks-service-role
Status              : CREATING
Version             : 1.10
```

- Pour API plus de détails, consultez la section CreateClusterRéférence des AWS Tools for PowerShell applets de commande.

### `New-EKSFargateProfile`

L'exemple de code suivant montre comment utiliser`New-EKSFargateProfile`.

Outils pour PowerShell

Exemple 1 : Cette applet de commande crée un profil Fargate pour AWS votre cluster Amazon. EKS Vous devez disposer d'au moins un profil Fargate dans un cluster pour pouvoir planifier des pods sur l'infrastructure Fargate.

```
New-EKSFargateProfile -FargateProfileName EKSFargateProfile -ClusterName TEST -
Subnet "subnet-02f6ff500ff2067a0", "subnet-0cd976f08d5fbfaae" -PodExecutionRoleArn
 arn:aws:iam::012345678912:role/AmazonEKSFargatePodExecutionRole -Selector
 @{Namespace="default"}
```

Sortie :

```
ClusterName         : TEST
CreatedAt           : 12/26/2019 12:38:21 PM
FargateProfileArn   : arn:aws:eks:us-east-2:012345678912:fargateprofile/TEST/
EKSFargateProfile/20b7a11b-8292-41c1-bc56-ffa5e60f6224
FargateProfileName  : EKSFargateProfile
PodExecutionRoleArn : arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole
Selectors           : {Amazon.EKS.Model.FargateProfileSelector}
Status              : CREATING
Subnets             : {subnet-0cd976f08d5fbfaae, subnet-02f6ff500ff2067a0}
Tags                : {}
```

- Pour API plus de détails, consultez la section CreateFargateProfileRéférence des AWS Tools for PowerShell applets de commande.

### `New-EKSNodeGroup`

L'exemple de code suivant montre comment utiliser`New-EKSNodeGroup`.

Outils pour PowerShell

Exemple 1 : Cette applet de commande crée un groupe de nœuds de travail gérés pour un cluster AmazonEKS. Vous pouvez uniquement créer un groupe de nœuds pour votre cluster qui soit égal à la version Kubernetes actuelle du cluster. Tous les groupes de nœuds sont créés avec la dernière AMI version de la version mineure correspondante de Kubernetes du cluster.

```
New-EKSNodeGroup -NodeGroupName "ProdEKSNodeGroup" -AmiType "AL2_x86_64"
 -DiskSize 40 -ClusterName "PROD" -ScalingConfig_DesiredSize 2 -
```

```
ScalingConfig_MinSize 2 -ScalingConfig_MaxSize 5 -InstanceType t3.large
 -NodeRole "arn:aws:iam::012345678912:role/NodeInstanceRole" -Subnet
 "subnet-0d1a9fff35efa7691","subnet-0a3f4928edbc224d4"
```

Sortie :

```
AmiType        : AL2_x86_64
ClusterName    : PROD
CreatedAt      : 12/25/2019 10:16:45 AM
DiskSize       : 40
Health         : Amazon.EKS.Model.NodegroupHealth
InstanceTypes  : {t3.large}
Labels         : {}
ModifiedAt     : 12/25/2019 10:16:45 AM
NodegroupArn   : arn:aws:eks:us-west-2:012345678912:nodegroup/PROD/
ProdEKSNodeGroup/7eb79e47-82b6-04d9-e984-95110db6fa85
NodegroupName  : ProdEKSNodeGroup
NodeRole       : arn:aws:iam::012345678912:role/NodeInstanceRole
ReleaseVersion : 1.14.7-20190927
RemoteAccess   :
Resources      :
ScalingConfig  : Amazon.EKS.Model.NodegroupScalingConfig
Status         : CREATING
Subnets        : {subnet-0d1a9fff35efa7691, subnet-0a3f4928edbc224d4}
Tags           : {}
Version        : 1.14
```

- Pour API plus de détails, consultez la section CreateNodegroupRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EKSCluster`

L'exemple de code suivant montre comment utiliser`Remove-EKSCluster`.

Outils pour PowerShell

Exemple 1 : Cette applet de commande supprime le plan de contrôle du EKS cluster Amazon.

```
Remove-EKSCluster -Name "DEV-KUBE-CL"
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSCluster (DeleteCluster)" on target "DEV-KUBE-
CL".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

Arn                  : arn:aws:eks:us-west-2:012345678912:cluster/DEV-KUBE-CL
CertificateAuthority : Amazon.EKS.Model.Certificate
ClientRequestToken   :
CreatedAt            : 12/25/2019 9:33:25 AM
Endpoint             : https://02E6D31E3E4F8C15D7BE7F58D527776A.yl4.us-
west-2.eks.amazonaws.com
Identity             : Amazon.EKS.Model.Identity
Logging              : Amazon.EKS.Model.Logging
Name                 : DEV-KUBE-CL
PlatformVersion      : eks.7
ResourcesVpcConfig   : Amazon.EKS.Model.VpcConfigResponse
RoleArn              : arn:aws:iam::012345678912:role/eks-iam-role
Status               : DELETING
Tags                 : {}
Version              : 1.14
```

- Pour API plus de détails, consultez la section DeleteClusterRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EKSFargateProfile`

L'exemple de code suivant montre comment utiliser`Remove-EKSFargateProfile`.

Outils pour PowerShell

Exemple 1 : cette applet de commande supprime un profil AWS Fargate. Lorsque vous supprimez un profil Fargate, tous les pods exécutés sur Fargate créés avec le profil sont supprimés.

```
Remove-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
```

```
Performing the operation "Remove-EKSFargateProfile (DeleteFargateProfile)" on target
 "EKSFargate".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

ClusterName         : TEST
CreatedAt           : 12/26/2019 12:34:47 PM
FargateProfileArn   : arn:aws:eks:us-east-2:012345678912:fargateprofile/TEST/
EKSFargate/42b7a119-e16b-a279-ce97-bdf303adec92
FargateProfileName  : EKSFargate
PodExecutionRoleArn : arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole
Selectors           : {Amazon.EKS.Model.FargateProfileSelector}
Status              : DELETING
Subnets             : {subnet-0cd976f08d5fbfaae, subnet-02f6ff500ff2067a0}
Tags                : {}
```

- Pour API plus de détails, consultez la section DeleteFargateProfileRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EKSNodegroup`

L'exemple de code suivant montre comment utiliser`Remove-EKSNodegroup`.

Outils pour PowerShell

Exemple 1 : Cette applet de commande supprime un groupe de EKS nœuds Amazon pour un cluster.

```
Remove-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSNodegroup (DeleteNodegroup)" on target
 "ProdEKSNodeGroup".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

AmiType        : AL2_x86_64
ClusterName    : PROD
```

```
CreatedAt      : 12/25/2019 10:16:45 AM
DiskSize       : 40
Health         : Amazon.EKS.Model.NodegroupHealth
InstanceTypes  : {t3.large}
Labels         : {}
ModifiedAt     : 12/25/2019 11:01:16 AM
NodegroupArn   : arn:aws:eks:us-west-2:012345678912:nodegroup/PROD/
ProdEKSNodeGroup/7eb79e47-82b6-04d9-e984-95110db6fa85
NodegroupName  : ProdEKSNodeGroup
NodeRole       : arn:aws:iam::012345678912:role/NodeInstanceRole
ReleaseVersion : 1.14.7-20190927
RemoteAccess   :
Resources      : Amazon.EKS.Model.NodegroupResources
ScalingConfig  : Amazon.EKS.Model.NodegroupScalingConfig
Status         : DELETING
Subnets        : {subnet-0d1a9fff35efa7691, subnet-0a3f4928edbc224d4}
Tags           : {}
Version        : 1.14
```

- Pour API plus de détails, consultez la section DeleteNodegroupRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-EKSResourceTag`

L'exemple de code suivant montre comment utiliser`Remove-EKSResourceTag`.

Outils pour PowerShell

Exemple 1 : cette applet de commande supprime les balises spécifiées d'une ressource. EKS

```
Remove-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
 -TagKey "Name"
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSResourceTag (UntagResource)" on target
 "arn:aws:eks:us-west-2:012345678912:cluster/PROD".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Pour API plus de détails, consultez la section UntagResourceRéférence des AWS Tools for PowerShell applets de commande.

### `Update-EKSClusterConfig`

L'exemple de code suivant montre comment utiliser`Update-EKSClusterConfig`.

Outils pour PowerShell

Exemple 1 : met à jour la configuration d'un EKS cluster Amazon. Votre cluster continue de fonctionner pendant la mise à jour.

```
Update-EKSClusterConfig -Name "PROD" -Logging_ClusterLogging
 @{Types="api","audit","authenticator","controllerManager","scheduler",Enabled="True"}
```

Sortie :

```
CreatedAt : 12/25/2019 5:03:07 PM
Errors    : {}
Id        : ee708232-7d2e-4ed7-9270-d0b5176f0726
Params    : {Amazon.EKS.Model.UpdateParam}
Status    : InProgress
Type      : LoggingUpdate
```

- Pour API plus de détails, consultez la section UpdateClusterConfigRéférence des AWS Tools for PowerShell applets de commande.

### `Update-EKSClusterVersion`

L'exemple de code suivant montre comment utiliser`Update-EKSClusterVersion`.

Outils pour PowerShell

Exemple 1 : Cette applet de commande met à jour un EKS cluster Amazon vers la version Kubernetes spécifiée. Votre cluster continue de fonctionner pendant la mise à jour.

```
Update-EKSClusterVersion -Name "PROD-KUBE-CL" -Version 1.14
```

Sortie :

```
CreatedAt : 12/26/2019 9:50:37 AM
Errors    : {}
Id        : ef186eff-3b3a-4c25-bcfc-3dcdf9e898a8
Params    : {Amazon.EKS.Model.UpdateParam, Amazon.EKS.Model.UpdateParam}
Status    : InProgress
Type      : VersionUpdate
```

- Pour API plus de détails, consultez la section UpdateClusterVersionRéférence des AWS Tools for PowerShell applets de commande.

# Elastic Load Balancing - Exemples de version 1 utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide de la AWS Tools for PowerShell version 1 d'Elastic Load Balancing.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `Add-ELBLoadBalancerToSubnet`

L'exemple de code suivant montre comment utiliser`Add-ELBLoadBalancerToSubnet`.

Outils pour PowerShell

Exemple 1 : Cet exemple ajoute le sous-réseau spécifié à l'ensemble de sous-réseaux configurés pour l'équilibreur de charge spécifié. La sortie inclut la liste complète des sous-réseaux.

```
Add-ELBLoadBalancerToSubnet -LoadBalancerName my-load-balancer -Subnet
 subnet-12345678
```

Sortie :

```
subnet-12345678
subnet-87654321
```

- Pour API plus de détails, consultez la section AttachLoadBalancerToSubnetsRéférence des AWS Tools for PowerShell applets de commande.

### `Add-ELBResourceTag`

L'exemple de code suivant montre comment utiliser`Add-ELBResourceTag`.

Outils pour PowerShell

Exemple 1 : Cet exemple ajoute les balises spécifiées à l'équilibreur de charge spécifié. La syntaxe utilisée dans cet exemple nécessite PowerShell la version 3 ou ultérieure.

```
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag
 @{ Key="project";Value="lima" },@{ Key="department";Value="digital-media" }
```

Exemple 2 : avec PowerShell la version 2, vous devez utiliser New-Object pour créer une balise pour le paramètre Tag.

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.Tag
$tag.Key = "project"
$tag.Value = "lima"
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag $tag
```

- Pour API plus de détails, consultez la section AddTagsRéférence des AWS Tools for PowerShell applets de commande.

### `Disable-ELBAvailabilityZoneForLoadBalancer`

L'exemple de code suivant montre comment utiliser`Disable-ELBAvailabilityZoneForLoadBalancer`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la zone de disponibilité spécifiée de l'équilibreur de charge spécifié. La sortie inclut les zones de disponibilité restantes.

```
Disable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -
AvailabilityZone us-west-2a
```

Sortie :

```
us-west-2b
```

- Pour API plus de détails, consultez la section DisableAvailabilityZonesForLoadBalancerRéférence des AWS Tools for PowerShell applets de commande.

### `Dismount-ELBLoadBalancerFromSubnet`

L'exemple de code suivant montre comment utiliser`Dismount-ELBLoadBalancerFromSubnet`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le sous-réseau spécifié de l'ensemble de sous-réseaux configurés pour l'équilibreur de charge spécifié. La sortie inclut les sous-réseaux restants.

```
Dismount-ELBLoadBalancerFromSubnet -LoadBalancerName my-load-balancer -Subnet
 subnet-12345678
```

Sortie :

```
subnet-87654321
```

- Pour API plus de détails, consultez la section DetachLoadBalancerFromSubnetsRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-ELBLoadBalancerAttribute`

L'exemple de code suivant montre comment utiliser`Edit-ELBLoadBalancerAttribute`.

Outils pour PowerShell

Exemple 1 : Cet exemple active l'équilibrage de charge entre zones pour l'équilibreur de charge spécifié.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -
CrossZoneLoadBalancing_Enabled $true
```

Exemple 2 : Cet exemple désactive le drainage des connexions pour l'équilibreur de charge spécifié.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -
ConnectionDraining_Enabled $false
```

Exemple 3 : Cet exemple active la journalisation des accès pour l'équilibreur de charge spécifié.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer `
>> -AccessLog_Enabled $true `
>> -AccessLog_S3BucketName amzn-s3-demo-logging-bucket `
>> -AccessLog_S3BucketPrefix my-app/prod `
>> -AccessLog_EmitInterval 60
```

- Pour API plus de détails, consultez la section ModifyLoadBalancerAttributesRéférence des AWS Tools for PowerShell applets de commande.

### `Enable-ELBAvailabilityZoneForLoadBalancer`

L'exemple de code suivant montre comment utiliser`Enable-ELBAvailabilityZoneForLoadBalancer`.

Outils pour PowerShell

Exemple 1 : Cet exemple ajoute la zone de disponibilité spécifiée à l'équilibreur de charge spécifié. La sortie inclut la liste complète des zones de disponibilité.

```
Enable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -
AvailabilityZone us-west-2a
```

Sortie :

```
us-west-2a
us-west-2b
```

- Pour API plus de détails, consultez la section [EnableAvailabilityZonesForLoadBalancer](#)Référence des AWS Tools for PowerShell applets de commande.

### `Get-ELBInstanceHealth`

L'exemple de code suivant montre comment utiliser`Get-ELBInstanceHealth`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit l'état des instances enregistrées auprès de l'équilibreur de charge spécifié.

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer
```

Sortie :

```
Description                  InstanceId                   ReasonCode
      State
-----------                  ----------                   ----------
      -----
N/A                          i-87654321                   N/A
      InService
Instance has failed at lea... i-12345678                  Instance
      OutOfService
```

Exemple 2 : Cet exemple décrit l'état de l'instance spécifiée enregistrée auprès de l'équilibreur de charge spécifié.

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance i-12345678
```

Exemple 3 : Cet exemple affiche la description complète de l'état de l'instance spécifiée.

```
(Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance
 i-12345678).Description
```

Sortie :

```
Instance has failed at least the UnhealthyThreshold number of health checks
 consecutively.
```

- Pour API plus de détails, consultez la section DescribeInstanceHealthRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ELBLoadBalancer`

L'exemple de code suivant montre comment utiliser`Get-ELBLoadBalancer`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie les noms de vos équilibreurs de charge.

```
Get-ELBLoadBalancer | format-table -property LoadBalancerName
```

Sortie :

```
LoadBalancerName
----------------
my-load-balancer
my-other-load-balancer
my-internal-load-balancer
```

Exemple 2 : Cet exemple décrit l'équilibreur de charge spécifié.

```
Get-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

Sortie :

```
AvailabilityZones         : {us-west-2a, us-west-2b}
BackendServerDescriptions :
 {Amazon.ElasticLoadBalancing.Model.BackendServerDescription}
CanonicalHostedZoneName   : my-load-balancer-1234567890.us-west-2.elb.amazonaws.com
CanonicalHostedZoneNameID : Z3DZXE0EXAMPLE
CreatedTime               : 4/11/2015 12:12:45 PM
DNSName                   : my-load-balancer-1234567890.us-west-2.elb.amazonaws.com
HealthCheck               : Amazon.ElasticLoadBalancing.Model.HealthCheck
```

```
Instances                 : {i-207d9717, i-afefb49b}
ListenerDescriptions      : {Amazon.ElasticLoadBalancing.Model.ListenerDescription}
LoadBalancerName          : my-load-balancer
Policies                  : Amazon.ElasticLoadBalancing.Model.Policies
Scheme                    : internet-facing
SecurityGroups            : {sg-a61988c3}
SourceSecurityGroup       : Amazon.ElasticLoadBalancing.Model.SourceSecurityGroup
Subnets                   : {subnet-15aaab61}
VPCId                     : vpc-a01106c2
```

Exemple 3 : Cet exemple décrit tous vos équilibreurs de charge dans la AWS région actuelle.

```
Get-ELBLoadBalancer
```

Exemple 4 : Cet exemple décrit tous vos équilibreurs de charge parmi tous ceux disponibles. Régions AWS

```
Get-AWSRegion | % { Get-ELBLoadBalancer -Region $_ }
```

- Pour API plus de détails, consultez la section DescribeLoadBalancersRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ELBLoadBalancerAttribute`

L'exemple de code suivant montre comment utiliser`Get-ELBLoadBalancerAttribute`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit les attributs de l'équilibreur de charge spécifié.

```
Get-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer
```

Sortie :

```
AccessLog              : Amazon.ElasticLoadBalancing.Model.AccessLog
AdditionalAttributes   : {}
ConnectionDraining     : Amazon.ElasticLoadBalancing.Model.ConnectionDraining
ConnectionSettings     : Amazon.ElasticLoadBalancing.Model.ConnectionSettings
CrossZoneLoadBalancing : Amazon.ElasticLoadBalancing.Model.CrossZoneLoadBalancing
```

- Pour API plus de détails, consultez la section DescribeLoadBalancerAttributesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ELBLoadBalancerPolicy`

L'exemple de code suivant montre comment utiliser`Get-ELBLoadBalancerPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit les politiques associées à l'équilibreur de charge spécifié.

```
Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer
```

Sortie :

```
PolicyAttributeDescriptions             PolicyName
 PolicyTypeName
----------------------------            ----------
 --------------
{ProxyProtocol}                         my-ProxyProtocol-policy
 ProxyProtocolPolicyType
{CookieName}                            my-app-cookie-policy
 AppCookieStickinessPolicyType
```

Exemple 2 : Cet exemple décrit les attributs de la politique spécifiée.

```
(Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
ProxyProtocol-policy).PolicyAttributeDescriptions
```

Sortie :

```
AttributeName    AttributeValue
-------------    --------------
ProxyProtocol    true
```

Exemple 3 : Cet exemple décrit les politiques prédéfinies, y compris les exemples de politiques. Les noms des exemples de politiques comportent le préfixe ELBSample -.

```
Get-ELBLoadBalancerPolicy
```

Sortie :

```
PolicyAttributeDescriptions             PolicyName
 PolicyTypeName
--------------------------             ----------
 --------------
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2015-05
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2015-03
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2015-02
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2014-10
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2014-01
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2011-08
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSample-ELBDefaultCipherPolicy
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSample-OpenSSLDefaultCipherPolicy
 SSLNegotiationPolicyType
```

- Pour API plus de détails, consultez la section DescribeLoadBalancerPoliciesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ELBLoadBalancerPolicyType`

L'exemple de code suivant montre comment utiliser`Get-ELBLoadBalancerPolicyType`.

Outils pour PowerShell

Exemple 1 : Cet exemple obtient les types de politiques pris en charge par Elastic Load Balancing.

```
Get-ELBLoadBalancerPolicyType
```

Sortie :

```
Description                             PolicyAttributeTypeDescriptions
 PolicyTypeName
```

```
-----------                              -------------------------------
 --------------
Stickiness policy with session lifet... {CookieExpirationPeriod}
 LBCookieStickinessPolicyType
Policy that controls authentication ... {PublicKeyPolicyName}
 BackendServerAuthenticationPolicyType
Listener policy that defines the cip... {Protocol-SSLv2, Protocol-TLSv1, Pro...
 SSLNegotiationPolicyType
Policy containing a list of public k... {PublicKey}
 PublicKeyPolicyType
Stickiness policy with session lifet... {CookieName}
 AppCookieStickinessPolicyType
Policy that controls whether to incl... {ProxyProtocol}
 ProxyProtocolPolicyType
```

Exemple 2 : Cet exemple décrit le type de politique spécifié.

```
Get-ELBLoadBalancerPolicyType -PolicyTypeName ProxyProtocolPolicyType
```

Sortie :

```
Description                              PolicyAttributeTypeDescriptions
 PolicyTypeName
-----------                              -------------------------------
 --------------
Policy that controls whether to incl... {ProxyProtocol}
 ProxyProtocolPolicyType
```

Exemple 3 : Cet exemple affiche la description complète du type de politique spécifié.

```
(Get-ELBLoadBalancerPolicyType -PolicyTypeName).Description
```

Sortie :

```
Policy that controls whether to include the IP address and port of the originating
 request for TCP messages.
This policy operates on TCP/SSL listeners only
```

- Pour API plus de détails, consultez la section DescribeLoadBalancerPolicyTypesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ELBResourceTag`

L'exemple de code suivant montre comment utiliser`Get-ELBResourceTag`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie les balises pour les équilibreurs de charge spécifiés.

```
Get-ELBResourceTag -LoadBalancerName @("my-load-balancer","my-internal-load-
balancer")
```

Sortie :

```
LoadBalancerName             Tags
----------------             ----
my-load-balancer             {project, department}
my-internal-load-balancer    {project, department}
```

Exemple 2 : Cet exemple décrit les balises de l'équilibreur de charge spécifié.

```
(Get-ELBResourceTag -LoadBalancerName my-load-balancer).Tags
```

Sortie :

```
Key           Value
---           -----
project       lima
department    digital-media
```

- Pour API plus de détails, consultez la section DescribeTagsRéférence des AWS Tools for PowerShell applets de commande.

### `Join-ELBSecurityGroupToLoadBalancer`

L'exemple de code suivant montre comment utiliser`Join-ELBSecurityGroupToLoadBalancer`.

Outils pour PowerShell

Exemple 1 : Cet exemple remplace le groupe de sécurité actuel pour l'équilibreur de charge spécifié par le groupe de sécurité spécifié.

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -
SecurityGroup sg-87654321
```

Sortie :

```
sg-87654321
```

Exemple 2 : pour conserver le groupe de sécurité actuel et spécifier un groupe de sécurité supplémentaire, spécifiez à la fois les groupes de sécurité existants et les nouveaux.

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -
SecurityGroup @("sg-12345678", "sg-87654321")
```

Sortie :

```
sg-12345678
sg-87654321
```

- Pour API plus de détails, consultez la section ApplySecurityGroupsToLoadBalancerRéférence des AWS Tools for PowerShell applets de commande.

### `New-ELBAppCookieStickinessPolicy`

L'exemple de code suivant montre comment utiliser`New-ELBAppCookieStickinessPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une politique de persistance qui suit les durées de vie de session persistantes du cookie généré par l'application spécifié.

```
New-ELBAppCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-
app-cookie-policy -CookieName my-app-cookie
```

- Pour API plus de détails, consultez la section CreateAppCookieStickinessPolicyRéférence des AWS Tools for PowerShell applets de commande.

### `New-ELBLBCookieStickinessPolicy`

L'exemple de code suivant montre comment utiliser`New-ELBLBCookieStickinessPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une politique de fidélisation dont la durée de vie des sessions est contrôlée par la période d'expiration spécifiée (en secondes).

```
New-ELBLBCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-
duration-cookie-policy -CookieExpirationPeriod 60
```

Exemple 2 : Cet exemple crée une politique de persistance dont la durée de vie des sessions est contrôlée par la durée de vie du navigateur (agent utilisateur).

```
New-ELBLBCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-
duration-cookie-policy
```

- Pour API plus de détails, consultez la section CreateLbCookieStickinessPolicyRéférence des AWS Tools for PowerShell applets de commande.

### `New-ELBLoadBalancer`

L'exemple de code suivant montre comment utiliser`New-ELBLoadBalancer`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un équilibreur de charge avec un HTTP écouteur dans un. VPC

```
$httpListener = New-Object Amazon.ElasticLoadBalancing.Model.Listener
$httpListener.Protocol = "http"
$httpListener.LoadBalancerPort = 80
$httpListener.InstanceProtocol = "http"
$httpListener.InstancePort = 80
New-ELBLoadBalancer -LoadBalancerName my-vpc-load-balancer -SecurityGroup sg-
a61988c3 -Subnet subnet-15aaab61 -Listener $httpListener

my-vpc-load-balancer-1234567890.us-west-2.elb.amazonaws.com
```

Exemple 2 : Cet exemple crée un équilibreur de charge avec un HTTP écouteur dans EC2 - Classic.

```
New-ELBLoadBalancer -LoadBalancerName my-classic-load-balancer -AvailabilityZone us-
west-2a` -Listener $httpListener
```

Sortie :

```
my-classic-load-balancer-123456789.us-west-2.elb.amazonaws.com
```

Exemple 3 : Cet exemple crée un équilibreur de charge avec un HTTPS écouteur.

```
$httpsListener = New-Object Amazon.ElasticLoadBalancing.Model.Listener
$httpsListener.Protocol = "https"
$httpsListener.LoadBalancerPort = 443
$httpsListener.InstanceProtocol = "http"
$httpsListener.InstancePort = 80
$httpsListener.SSLCertificateId="arn:aws:iam::123456789012:server-certificate/my-
server-cert"
New-ELBLoadBalancer -LoadBalancerName my-load-balancer -AvailabilityZone us-west-2a
 -Listener $httpsListener

my-load-balancer-123456789.us-west-2.elb.amazonaws.com
```

- Pour API plus de détails, consultez la section CreateLoadBalancerRéférence des AWS Tools for PowerShell applets de commande.

### `New-ELBLoadBalancerListener`

L'exemple de code suivant montre comment utiliser`New-ELBLoadBalancerListener`.

Outils pour PowerShell

Exemple 1 : cet exemple ajoute un HTTPS écouteur à l'équilibreur de charge spécifié.

```
$httpsListener = New-Object Amazon.ElasticLoadBalancing.Model.Listener
$httpsListener.Protocol = "https"
$httpsListener.LoadBalancerPort = 443
$httpsListener.InstanceProtocol = "https"
$httpsListener.InstancePort = 443
$httpsListener.SSLCertificateId="arn:aws:iam::123456789012:server-certificate/my-
server-cert"
New-ELBLoadBalancerListener -LoadBalancerName my-load-balancer -Listener
 $httpsListener
```

- Pour API plus de détails, consultez la section CreateLoadBalancerListenersRéférence des AWS Tools for PowerShell applets de commande.

### `New-ELBLoadBalancerPolicy`

L'exemple de code suivant montre comment utiliser`New-ELBLoadBalancerPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une nouvelle politique de protocole proxy pour un équilibreur de charge spécifié.

```
$attribute = New-Object Amazon.ElasticLoadBalancing.Model.PolicyAttribute -Property
 @{
         AttributeName="ProxyProtocol"
         AttributeValue="True"
    }
New-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
ProxyProtocol-policy -PolicyTypeName ProxyProtocolPolicyType -PolicyAttribute
 $attribute
```

- Pour API plus de détails, consultez la section CreateLoadBalancerPolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Register-ELBInstanceWithLoadBalancer`

L'exemple de code suivant montre comment utiliser`Register-ELBInstanceWithLoadBalancer`.

Outils pour PowerShell

Exemple 1 : Cet exemple enregistre l'EC2instance spécifiée auprès de l'équilibreur de charge spécifié.

```
Register-ELBInstanceWithLoadBalancer -LoadBalancerName my-load-balancer -Instance
 i-12345678
```

Sortie :

```
InstanceId
----------
i-12345678
i-87654321
```

- Pour API plus de détails, consultez la section RegisterInstancesWithLoadBalancerRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-ELBInstanceFromLoadBalancer`

L'exemple de code suivant montre comment utiliser`Remove-ELBInstanceFromLoadBalancer`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime l'EC2instance spécifiée de l'équilibreur de charge spécifié. Vous êtes invité à confirmer avant de poursuivre l'opération, sauf si vous spécifiez également le paramètre Force.

```
Remove-ELBInstanceFromLoadBalancer -LoadBalancerName my-load-balancer -Instance
 i-12345678
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBInstanceFromLoadBalancer
 (DeregisterInstancesFromLoadBalancer)" on Target
"Amazon.ElasticLoadBalancing.Model.Instance".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):

InstanceId
----------
i-87654321
```

- Pour API plus de détails, consultez la section DeregisterInstancesFromLoadBalancerRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-ELBLoadBalancer`

L'exemple de code suivant montre comment utiliser`Remove-ELBLoadBalancer`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime l'équilibreur de charge spécifié. Vous êtes invité à confirmer avant de poursuivre l'opération, sauf si vous spécifiez également le paramètre Force.

```
Remove-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBLoadBalancer (DeleteLoadBalancer)" on Target "my-
load-balancer".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Pour API plus de détails, consultez la section DeleteLoadBalancerRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-ELBLoadBalancerListener`

L'exemple de code suivant montre comment utiliser`Remove-ELBLoadBalancerListener`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime l'écouteur sur le port 80 pour l'équilibreur de charge spécifié. Vous êtes invité à confirmer avant de poursuivre l'opération, sauf si vous spécifiez également le paramètre Force.

```
Remove-ELBLoadBalancerListener -LoadBalancerName my-load-balancer -LoadBalancerPort
 80
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBLoadBalancerListener (DeleteLoadBalancerListeners)"
 on Target "80".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Pour API plus de détails, consultez la section DeleteLoadBalancerListenersRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-ELBLoadBalancerPolicy`

L'exemple de code suivant montre comment utiliser`Remove-ELBLoadBalancerPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la politique spécifiée de l'équilibreur de charge spécifié. Vous êtes invité à confirmer avant de poursuivre l'opération, sauf si vous spécifiez également le paramètre Force.

```
Remove-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
duration-cookie-policy
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBLoadBalancerPolicy (DeleteLoadBalancerPolicy)" on
 Target "my-duration-cookie-policy".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Pour API plus de détails, consultez la section DeleteLoadBalancerPolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-ELBResourceTag`

L'exemple de code suivant montre comment utiliser`Remove-ELBResourceTag`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la balise spécifiée de l'équilibreur de charge spécifié. Vous êtes invité à confirmer avant de poursuivre l'opération, sauf si vous spécifiez également le paramètre Force. La syntaxe utilisée dans cet exemple nécessite PowerShell la version 3 ou ultérieure.

```
Remove-ELBResourceTag -LoadBalancerName my-load-balancer -Tag @{ Key="project" }
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELBResourceTag (RemoveTags)" on target
 "Amazon.ElasticLoadBalancing.Model.TagKeyOnly".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemple 2 : avec Powershell version 2, vous devez utiliser New-Object pour créer la balise pour le paramètre Tag.

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.TagKeyOnly
$tag.Key = "project"
Remove-ELBResourceTag -Tag $tag -Force
```

- Pour API plus de détails, consultez la section RemoveTagsRéférence des AWS Tools for PowerShell applets de commande.

### `Set-ELBHealthCheck`

L'exemple de code suivant montre comment utiliser`Set-ELBHealthCheck`.

Outils pour PowerShell

Exemple 1 : Cet exemple configure les paramètres de vérification de l'état de l'équilibreur de charge spécifié.

```
Set-ELBHealthCheck -LoadBalancerName my-load-balancer `
>> -HealthCheck_HealthyThreshold 2 `
>> -HealthCheck_UnhealthyThreshold 2 `
>> -HealthCheck_Target "HTTP:80/ping" `
>> -HealthCheck_Interval 30 `
>> -HealthCheck_Timeout 3
```

Sortie :

```
HealthyThreshold   : 2
Interval           : 30
Target             : HTTP:80/ping
Timeout            : 3
UnhealthyThreshold : 2
```

- Pour API plus de détails, consultez la section ConfigureHealthCheckRéférence des AWS Tools for PowerShell applets de commande.

### `Set-ELBLoadBalancerListenerSSLCertificate`

L'exemple de code suivant montre comment utiliser`Set-ELBLoadBalancerListenerSSLCertificate`.

Outils pour PowerShell

Exemple 1 : Cet exemple remplace le certificat qui met fin aux SSL connexions pour l'écouteur spécifié.

```
Set-ELBLoadBalancerListenerSSLCertificate -LoadBalancerName my-load-balancer `
>> -LoadBalancerPort 443 `
>> -SSLCertificateId "arn:aws:iam::123456789012:server-certificate/new-server-cert"
```

- Pour API plus de détails, consultez la section SetLoadBalancerListenerSslCertificateRéférence des AWS Tools for PowerShell applets de commande.

### `Set-ELBLoadBalancerPolicyForBackendServer`

L'exemple de code suivant montre comment utiliser`Set-ELBLoadBalancerPolicyForBackendServer`.

Outils pour PowerShell

Exemple 1 : Cet exemple remplace les politiques du port spécifié par la politique spécifiée.

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -
InstancePort 80 -PolicyName my-ProxyProtocol-policy
```

Exemple 2 : Cet exemple supprime toutes les politiques associées au port spécifié.

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -
InstancePort 80
```

- Pour API plus de détails, consultez la section SetLoadBalancerPoliciesForBackendServerRéférence des AWS Tools for PowerShell applets de commande.

### `Set-ELBLoadBalancerPolicyOfListener`

L'exemple de code suivant montre comment utiliser`Set-ELBLoadBalancerPolicyOfListener`.

Outils pour PowerShell

Exemple 1 : Cet exemple remplace les politiques de l'écouteur spécifié par la politique spécifiée.

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 443 -PolicyName my-SSLNegotiation-policy
```

Exemple 2 : Cet exemple supprime toutes les politiques associées à l'écouteur spécifié.

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 443
```

- Pour API plus de détails, consultez la section SetLoadBalancerPoliciesOfListenerRéférence des AWS Tools for PowerShell applets de commande.

## Elastic Load Balancing - Exemples de version 2 utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide de la AWS Tools for PowerShell version 2 d'Elastic Load Balancing.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

# Actions

### `Add-ELB2ListenerCertificate`

L'exemple de code suivant montre comment utiliser`Add-ELB2ListenerCertificate`.

Outils pour PowerShell

Exemple 1 : Cet exemple ajoute un certificat supplémentaire au récepteur spécifié.

```
Add-ELB2ListenerCertificate -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618' -
Certificate @{CertificateArn = 'arn:aws:acm:us-
east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97'}
```

Sortie :

```
CertificateArn
 IsDefault
--------------
 ---------
arn:aws:acm:us-east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97
 False
```

- Pour API plus de détails, consultez la section AddListenerCertificatesRéférence des AWS Tools for PowerShell applets de commande.

### `Add-ELB2Tag`

L'exemple de code suivant montre comment utiliser`Add-ELB2Tag`.

Outils pour PowerShell

Exemple 1 : Cet exemple ajoute un nouveau tag à la **`AWS.Tools.ElasticLoadBalancingV2`** ressource spécifiée.

```
Add-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Tag @{Key =
 'productVersion'; Value = '1.0.0'}
```

- Pour API plus de détails, consultez la section AddTagsRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-ELB2Listener`

L'exemple de code suivant montre comment utiliser`Edit-ELB2Listener`.

Outils pour PowerShell

Exemple 1 : Cet exemple modifie l'action par défaut des auditeurs spécifiés en réponse fixe.

```
$newDefaultAction = [Amazon.ElasticLoadBalancingV2.Model.Action]@{
    "FixedResponseConfig" = @{
    "ContentType" = "text/plain"
    "MessageBody" = "Hello World"
    "StatusCode" = "200"
  }
  "Type" = [Amazon.ElasticLoadBalancingV2.ActionTypeEnum]::FixedResponse
}

Edit-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/testALB/3e2f03b558e19676/d19f2f14974db685' -Port
 8080 -DefaultAction $newDefaultAction
```

Sortie :

```
Certificates    : {}
DefaultActions  : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn     : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/
testALB/3e2f03b558e19676/d19f2f14974db685
LoadBalancerArn : arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/
app/testALB/3e2f03b558e19676
Port            : 8080
Protocol        : HTTP
SslPolicy       :
```

- Pour API plus de détails, consultez la section ModifyListenerRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-ELB2LoadBalancerAttribute`

L'exemple de code suivant montre comment utiliser`Edit-ELB2LoadBalancerAttribute`.

Outils pour PowerShell

Exemple 1 : Cet exemple modifie les attributs de l'équilibreur de charge spécifié.

```
Edit-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Attribute @{Key =
 'deletion_protection.enabled'; Value = 'true'}
```

Sortie :

```
Key                                             Value
---                                             -----
deletion_protection.enabled                     true
access_logs.s3.enabled                          false
access_logs.s3.bucket
access_logs.s3.prefix
idle_timeout.timeout_seconds                    60
routing.http2.enabled                           true
routing.http.drop_invalid_header_fields.enabled false
```

- Pour API plus de détails, consultez la section ModifyLoadBalancerAttributesRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-ELB2Rule`

L'exemple de code suivant montre comment utiliser`Edit-ELB2Rule`.

Outils pour PowerShell

Exemple 1 : Cet exemple modifie les configurations de règles d'écoute spécifiées.

```
$newRuleCondition = [Amazon.ElasticLoadBalancingV2.Model.RuleCondition]@{
    "PathPatternConfig" = @{
    "Values" = "/login1","/login2","/login3"
  }
  "Field" = "path-pattern"
}
```

```
Edit-ELB2Rule -RuleArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener-rule/app/testALB/3e2f03b558e19676/1c84f02aec143e80/
f4f51dfaa033a8cc' -Condition $newRuleCondition
```

Sortie :

```
Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 10
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/
testALB/3e2f03b558e19676/1c84f02aec143e80/f4f51dfaa033a8cc
```

- Pour API plus de détails, consultez la section ModifyRuleRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-ELB2TargetGroup`

L'exemple de code suivant montre comment utiliser`Edit-ELB2TargetGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple modifie les propriétés du groupe cible spécifié.

```
Edit-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -HealthCheckIntervalSecond
 60 -HealthCheckPath '/index.html' -HealthCheckPort 8080
```

Sortie :

```
HealthCheckEnabled         : True
HealthCheckIntervalSeconds : 60
HealthCheckPath            : /index.html
HealthCheckPort            : 8080
HealthCheckProtocol        : HTTP
HealthCheckTimeoutSeconds  : 5
HealthyThresholdCount      : 5
LoadBalancerArns           : {}
Matcher                    : Amazon.ElasticLoadBalancingV2.Model.Matcher
Port                       : 80
```

```
Protocol                    : HTTP
TargetGroupArn              : arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970
TargetGroupName             : test-tg
TargetType                  : instance
UnhealthyThresholdCount     : 2
VpcId                       : vpc-2cfd7000
```

- Pour API plus de détails, consultez la section ModifyTargetGroupRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-ELB2TargetGroupAttribute`

L'exemple de code suivant montre comment utiliser`Edit-ELB2TargetGroupAttribute`.

Outils pour PowerShell

Exemple 1 : Cet exemple modifie l'attribut deregistration_delay du groupe cible spécifié.

```
Edit-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Attribute @{Key =
 'deregistration_delay.timeout_seconds'; Value = 600}
```

Sortie :

```
Key                                 Value
---                                 -----
stickiness.enabled                  false
deregistration_delay.timeout_seconds  600
stickiness.type                     lb_cookie
stickiness.lb_cookie.duration_seconds 86400
slow_start.duration_seconds         0
load_balancing.algorithm.type       round_robin
```

- Pour API plus de détails, consultez la section ModifyTargetGroupAttributesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ELB2AccountLimit`

L'exemple de code suivant montre comment utiliser`Get-ELB2AccountLimit`.

Outils pour PowerShell

Exemple 1 : Cette commande répertorie les limites de ELB2 compte pour une région donnée.

```
Get-ELB2AccountLimit
```

Sortie :

```
Max  Name
---  ----
3000 target-groups
1000 targets-per-application-load-balancer
50   listeners-per-application-load-balancer
100  rules-per-application-load-balancer
50   network-load-balancers
3000 targets-per-network-load-balancer
500  targets-per-availability-zone-per-network-load-balancer
50   listeners-per-network-load-balancer
5    condition-values-per-alb-rule
5    condition-wildcards-per-alb-rule
100  target-groups-per-application-load-balancer
5    target-groups-per-action-on-application-load-balancer
1    target-groups-per-action-on-network-load-balancer
50   application-load-balancers
```

- Pour API plus de détails, consultez la section DescribeAccountLimitsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ELB2Listener`

L'exemple de code suivant montre comment utiliser`Get-ELB2Listener`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit les écouteurs duALB/NLBspécifié.

```
Get-ELB2Listener -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Sortie :

```
Certificates    : {}
```

```
DefaultActions  : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn     : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/
test-alb/3651b4394dd9a24f/1dac07c21187d41e
LoadBalancerArn : arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/
app/test-alb/3651b4394dd9a24f
Port            : 80
Protocol        : HTTP
SslPolicy       :

Certificates    : {Amazon.ElasticLoadBalancingV2.Model.Certificate}
DefaultActions  : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn     : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/
test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b
LoadBalancerArn : arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/
app/test-alb/3651b4394dd9a24f
Port            : 443
Protocol        : HTTPS
SslPolicy       : ELBSecurityPolicy-2016-08
```

- Pour API plus de détails, consultez la section DescribeListenersRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ELB2ListenerCertificate`

L'exemple de code suivant montre comment utiliser`Get-ELB2ListenerCertificate`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit le certificat de l'écouteur spécifié.

```
Get-ELB2ListenerCertificate -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Sortie :

```
CertificateArn
 IsDefault
--------------
 ---------
arn:aws:acm:us-east-1:123456789012:certificate/5fc7c092-68bf-4862-969c-22fd48b6e17c
 True
```

- Pour API plus de détails, consultez la section DescribeListenerCertificatesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ELB2LoadBalancer`

L'exemple de code suivant montre comment utiliser`Get-ELB2LoadBalancer`.

Outils pour PowerShell

Exemple 1 : Cet exemple affiche tous les équilibreurs de charge pour la région donnée.

```
Get-ELB2LoadBalancer
```

Sortie :

```
AvailabilityZones     : {us-east-1c}
CanonicalHostedZoneId : Z26RNL4JYFTOTI
CreatedTime           : 6/22/18 11:21:50 AM
DNSName               : test-elb1234567890-238d34ad8d94bc2e.elb.us-
east-1.amazonaws.com
IpAddressType         : ipv4
LoadBalancerArn       : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/net/test-elb1234567890/238d34ad8d94bc2e
LoadBalancerName      : test-elb1234567890
Scheme                : internet-facing
SecurityGroups        : {}
State                 : Amazon.ElasticLoadBalancingV2.Model.LoadBalancerState
Type                  : network
VpcId                 : vpc-2cf00000
```

- Pour API plus de détails, consultez la section DescribeLoadBalancersRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ELB2LoadBalancerAttribute`

L'exemple de code suivant montre comment utiliser`Get-ELB2LoadBalancerAttribute`.

Outils pour PowerShell

Exemple 1 : Cette commande décrit les attributs d'un équilibreur de charge donné.

```
Get-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/net/test-elb/238d34ad8d94bc2e'
```

Sortie :

```
Key                              Value
---                              -----
access_logs.s3.enabled           false
load_balancing.cross_zone.enabled true
access_logs.s3.prefix
deletion_protection.enabled      false
access_logs.s3.bucket
```

- Pour API plus de détails, consultez la section DescribeLoadBalancerAttributesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ELB2Rule`

L'exemple de code suivant montre comment utiliser`Get-ELB2Rule`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit les règles du récepteur pour le récepteur ARN spécifié.

```
Get-ELB2Rule -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Sortie :

```
Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 1
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/
test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b/2286fff5055e0f79

Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 2
```

```
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/
test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b/14e7b036567623ba

Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {}
IsDefault  : True
Priority   : default
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/
test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b/853948cf3aa9b2bf
```

- Pour API plus de détails, consultez la section DescribeRulesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ELB2SSLPolicy`

L'exemple de code suivant montre comment utiliser`Get-ELB2SSLPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie toutes les politiques d'écoute disponibles pour la ElasticLoadBalancing version 2.

```
Get-ELB2SSLPolicy
```

Sortie :

```
Ciphers
                              Name                                    SslProtocols
-------
                              ----                                    ------------
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-2016-08             {TLSv1,
 TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-2-2017-01     {TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-1-2017-01     {TLSv1.1,
 TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-2-Ext-2018-06 {TLSv1.2}
```

```
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-2018-06          {TLSv1,
 TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-2015-05             {TLSv1,
 TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-0-2015-04     {TLSv1,
 TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-1-2-Res-2019-08  {TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-1-1-2019-08      {TLSv1.1,
 TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-1-2-2019-08      {TLSv1.2}
```

- Pour API plus de détails, consultez la section DescribeSslPoliciesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ELB2Tag`

L'exemple de code suivant montre comment utiliser`Get-ELB2Tag`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie les balises de la ressource spécifiée.

```
Get-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Sortie :

```
ResourceArn
           Tags
-----------
           ----
arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f {stage, internalName, version}
```

- Pour API plus de détails, consultez la section DescribeTagsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ELB2TargetGroup`

L'exemple de code suivant montre comment utiliser`Get-ELB2TargetGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit le groupe cible spécifié.

```
Get-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Sortie :

```
HealthCheckEnabled         : True
HealthCheckIntervalSeconds : 30
HealthCheckPath            : /
HealthCheckPort            : traffic-port
HealthCheckProtocol        : HTTP
HealthCheckTimeoutSeconds  : 5
HealthyThresholdCount      : 5
LoadBalancerArns           : {arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f}
Matcher                    : Amazon.ElasticLoadBalancingV2.Model.Matcher
Port                       : 80
Protocol                   : HTTP
TargetGroupArn             : arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970
TargetGroupName            : test-tg
TargetType                 : instance
UnhealthyThresholdCount    : 2
VpcId                      : vpc-2cfd7000
```

- Pour API plus de détails, consultez la section DescribeTargetGroupsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ELB2TargetGroupAttribute`

L'exemple de code suivant montre comment utiliser`Get-ELB2TargetGroupAttribute`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit les attributs du groupe cible spécifié.

```
Get-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Sortie :

```
Key                                Value
---                                -----
stickiness.enabled                 false
deregistration_delay.timeout_seconds 300
stickiness.type                    lb_cookie
stickiness.lb_cookie.duration_seconds 86400
slow_start.duration_seconds        0
load_balancing.algorithm.type      round_robin
```

- Pour API plus de détails, consultez la section DescribeTargetGroupAttributesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ELB2TargetHealth`

L'exemple de code suivant montre comment utiliser`Get-ELB2TargetHealth`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie l'état de santé des cibles présentes dans le groupe cible spécifié.

```
Get-ELB2TargetHealth -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Sortie :

```
HealthCheckPort Target                                              TargetHealth
--------------- ------                                              ------------
80              Amazon.ElasticLoadBalancingV2.Model.TargetDescription
 Amazon.ElasticLoadBalancingV2.Model.TargetHealth
```

- Pour API plus de détails, consultez la section DescribeTargetHealthRéférence des AWS Tools for PowerShell applets de commande.

### `New-ELB2Listener`

L'exemple de code suivant montre comment utiliser`New-ELB2Listener`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un nouvel ALB écouteur avec l'action par défaut « Forward » pour envoyer le trafic au groupe cible spécifié.

```
$defaultAction = [Amazon.ElasticLoadBalancingV2.Model.Action]@{
  ForwardConfig = @{
    TargetGroups = @(
      @{ TargetGroupArn = "arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/testAlbTG/3d61c2f20aa5bccb" }
    )
    TargetGroupStickinessConfig = @{
      DurationSeconds = 900
      Enabled = $true
    }
  }
  Type = "Forward"
}

New-ELB2Listener -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/testALB/3e2f03b558e19676' -Port 8001 -Protocol
 "HTTP" -DefaultAction $defaultAction
```

Sortie :

```
Certificates    : {}
DefaultActions  : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn     : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/
testALB/3e2f03b558e19676/1c84f02aec143e80
LoadBalancerArn : arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/
app/testALB/3e2f03b558e19676
Port            : 8001
Protocol        : HTTP
SslPolicy       :
```

- Pour API plus de détails, consultez la section CreateListenerRéférence des AWS Tools for PowerShell applets de commande.

### `New-ELB2LoadBalancer`

L'exemple de code suivant montre comment utiliser`New-ELB2LoadBalancer`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un nouvel équilibreur de charge d'application connecté à Internet avec deux sous-réseaux.

```
New-ELB2LoadBalancer -Type application -Scheme internet-facing -IpAddressType
 ipv4 -Name 'New-Test-ALB' -SecurityGroup 'sg-07c3414abb8811cbd' -subnet 'subnet-
c37a67a6','subnet-fc02eea0'
```

Sortie :

```
AvailabilityZones     : {us-east-1b, us-east-1a}
CanonicalHostedZoneId : Z35SXDOTRQ7X7K
CreatedTime           : 12/28/19 2:58:03 PM
DNSName               : New-Test-ALB-1391502222.us-east-1.elb.amazonaws.com
IpAddressType         : ipv4
LoadBalancerArn       : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/New-Test-ALB/dab2e4d90eb51493
LoadBalancerName      : New-Test-ALB
Scheme                : internet-facing
SecurityGroups        : {sg-07c3414abb8811cbd}
State                 : Amazon.ElasticLoadBalancingV2.Model.LoadBalancerState
Type                  : application
VpcId                 : vpc-2cfd7000
```

- Pour API plus de détails, consultez la section CreateLoadBalancerRéférence des AWS Tools for PowerShell applets de commande.

### `New-ELB2Rule`

L'exemple de code suivant montre comment utiliser`New-ELB2Rule`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une nouvelle règle d'écouteur avec une action à réponse fixe basée sur la valeur d'en-tête du client pour le récepteur spécifié.

```
$newRuleAction = [Amazon.ElasticLoadBalancingV2.Model.Action]@{
```

```
  "FixedResponseConfig" = @{
    "ContentType" = "text/plain"
    "MessageBody" = "Hello World"
    "StatusCode" = "200"
  }
  "Type" = [Amazon.ElasticLoadBalancingV2.ActionTypeEnum]::FixedResponse
}

$newRuleCondition = [Amazon.ElasticLoadBalancingV2.Model.RuleCondition]@{
  "httpHeaderConfig" = @{
    "HttpHeaderName" = "customHeader"
    "Values" = "header2","header1"
  }
  "Field" = "http-header"
}

New-ELB2Rule -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/testALB/3e2f03b558e19676/1c84f02aec143e80' -Action
 $newRuleAction -Condition $newRuleCondition -Priority 10
```

Sortie :

```
Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 10
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/
testALB/3e2f03b558e19676/1c84f02aec143e80/f4f51dfaa033a8cc
```

- Pour API plus de détails, consultez la section CreateRuleRéférence des AWS Tools for PowerShell applets de commande.

### `New-ELB2TargetGroup`

L'exemple de code suivant montre comment utiliser`New-ELB2TargetGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un nouveau groupe cible avec les paramètres fournis.

```
New-ELB2TargetGroup -HealthCheckEnabled 1 -HealthCheckIntervalSeconds 30 -
HealthCheckPath '/index.html' -HealthCheckPort 80 -HealthCheckTimeoutSecond 5 -
```

```
HealthyThresholdCount 2 -UnhealthyThresholdCount 5 -Port 80 -Protocol 'HTTP' -
TargetType instance -VpcId 'vpc-2cfd7000' -Name 'NewTargetGroup'
```

Sortie :

```
HealthCheckEnabled         : True
HealthCheckIntervalSeconds : 30
HealthCheckPath            : /index.html
HealthCheckPort            : 80
HealthCheckProtocol        : HTTP
HealthCheckTimeoutSeconds  : 5
HealthyThresholdCount      : 2
LoadBalancerArns           : {}
Matcher                    : Amazon.ElasticLoadBalancingV2.Model.Matcher
Port                       : 80
Protocol                   : HTTP
TargetGroupArn             : arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/NewTargetGroup/534e484681d801bf
TargetGroupName            : NewTargetGroup
TargetType                 : instance
UnhealthyThresholdCount    : 5
VpcId                      : vpc-2cfd7000
```

- Pour API plus de détails, consultez la section CreateTargetGroupRéférence des AWS Tools for PowerShell applets de commande.

### `Register-ELB2Target`

L'exemple de code suivant montre comment utiliser`Register-ELB2Target`.

Outils pour PowerShell

Exemple 1 : Cet exemple enregistre l'instance « i-0672a4c4cdeae3111 » auprès du groupe cible spécifié.

```
Register-ELB2Target -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Target @{Port = 80; Id =
 'i-0672a4c4cdeae3111'}
```

- Pour API plus de détails, consultez la section RegisterTargetsRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-ELB2Listener`

L'exemple de code suivant montre comment utiliser`Remove-ELB2Listener`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le récepteur spécifié.

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Listener (DeleteListener)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-
alb/3651b4394dd9a24f/66e10e3aaf5b6d9b".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

Exemple 2 : Cet exemple supprime l'écouteur spécifié de l'équilibreur de charge.

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618'
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Listener (DeleteListener)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Pour API plus de détails, consultez la section DeleteListenerRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-ELB2ListenerCertificate`

L'exemple de code suivant montre comment utiliser`Remove-ELB2ListenerCertificate`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le certificat spécifié du groupe cible spécifié.

```
Remove-ELB2ListenerCertificate -Certificate @{CertificateArn = 'arn:aws:acm:us-
east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97'} -ListenerArn
 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618'
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2ListenerCertificate
 (RemoveListenerCertificates)" on target "arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Pour API plus de détails, consultez la section RemoveListenerCertificatesRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-ELB2LoadBalancer`

L'exemple de code suivant montre comment utiliser`Remove-ELB2LoadBalancer`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime l'équilibreur de charge spécifié.

```
Remove-ELB2LoadBalancer -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
```

```
Performing the operation "Remove-ELB2LoadBalancer (DeleteLoadBalancer)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Pour API plus de détails, consultez la section DeleteLoadBalancerRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-ELB2Rule`

L'exemple de code suivant montre comment utiliser`Remove-ELB2Rule`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la règle spécifiée du récepteur

```
Remove-ELB2Rule -RuleArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618/4b25eb10a42e33ab'
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Rule (DeleteRule)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618/4b25eb10a42e33ab".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Pour API plus de détails, consultez la section DeleteRuleRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-ELB2Tag`

L'exemple de code suivant montre comment utiliser`Remove-ELB2Tag`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la balise associée à la clé spécifiée.

```
Remove-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -TagKey
 'productVersion'
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Tag (RemoveTags)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Pour API plus de détails, consultez la section RemoveTagsRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-ELB2TargetGroup`

L'exemple de code suivant montre comment utiliser`Remove-ELB2TargetGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le groupe cible spécifié.

```
Remove-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/testssss/4e0b6076bc6483a7'
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2TargetGroup (DeleteTargetGroup)" on
 target "arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/
testssss/4e0b6076bc6483a7".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Pour API plus de détails, consultez la section DeleteTargetGroupRéférence des AWS Tools for PowerShell applets de commande.

### `Set-ELB2IpAddressType`

L'exemple de code suivant montre comment utiliser`Set-ELB2IpAddressType`.

Outils pour PowerShell

Exemple 1 : Cet exemple change le type d'adresse IP de l'équilibreur de charge de « IPv4 » à « DualStack ».

```
Set-ELB2IpAddressType -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -IpAddressType
 dualstack
```

Sortie :

```
Value
-----
dualstack
```

- Pour API plus de détails, consultez la section SetIpAddressTypeRéférence des AWS Tools for PowerShell applets de commande.

### `Set-ELB2RulePriority`

L'exemple de code suivant montre comment utiliser`Set-ELB2RulePriority`.

Outils pour PowerShell

Exemple 1 : Cet exemple modifie la priorité de la règle d'écoute spécifiée.

```
Set-ELB2RulePriority -RulePriority -RulePriority @{Priority = 11; RuleArn =
 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/a4eb199fa5046f80/dbf4c6dcef3ec6f8'}
```

Sortie :

```
Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 11
```

```
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/
test-alb/3651b4394dd9a24f/a4eb199fa5046f80/dbf4c6dcef3ec6f8
```

- Pour API plus de détails, consultez la section SetRulePrioritiesRéférence des AWS Tools for PowerShell applets de commande.

### `Set-ELB2SecurityGroup`

L'exemple de code suivant montre comment utiliser`Set-ELB2SecurityGroup`.

Outils pour PowerShell

Exemple 1 : cet exemple ajoute le groupe de sécurité « sg-07c3414abb8811cbd » à l'équilibreur de charge spécifié.

```
Set-ELB2SecurityGroup -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -SecurityGroup
 'sg-07c3414abb8811cbd'
```

Sortie :

```
sg-07c3414abb8811cbd
```

- Pour API plus de détails, consultez la section SetSecurityGroupsRéférence des AWS Tools for PowerShell applets de commande.

### `Set-ELB2Subnet`

L'exemple de code suivant montre comment utiliser`Set-ELB2Subnet`.

Outils pour PowerShell

Exemple 1 : Cet exemple modifie les sous-réseaux de l'équilibreur de charge spécifié.

```
Set-ELB2Subnet -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Subnet
 'subnet-7d8a0a51','subnet-c37a67a6'
```

Sortie :

```
LoadBalancerAddresses SubnetId        ZoneName
```

```
-------------------- --------        --------
{}                   subnet-7d8a0a51 us-east-1c
{}                   subnet-c37a67a6 us-east-1b
```

- Pour API plus de détails, consultez la section SetSubnetsRéférence des AWS Tools for PowerShell applets de commande.

### `Unregister-ELB2Target`

L'exemple de code suivant montre comment utiliser`Unregister-ELB2Target`.

Outils pour PowerShell

Exemple 1 : Cet exemple désenregistre l'instance « i-0672a4c4cdeae3111 » du groupe cible spécifié.

```
$targetDescription = New-Object
 Amazon.ElasticLoadBalancingV2.Model.TargetDescription
$targetDescription.Id = 'i-0672a4c4cdeae3111'
Unregister-ELB2Target -Target $targetDescription -TargetGroupArn
 'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/test-tg/
a4e04b3688be1970'
```

- Pour API plus de détails, consultez la section DeregisterTargetsRéférence des AWS Tools for PowerShell applets de commande.

# FSxExemples Amazon utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS Tools for PowerShell aide d'AmazonFSx.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

# Actions

### `Add-FSXResourceTag`

L'exemple de code suivant montre comment utiliser`Add-FSXResourceTag`.

Outils pour PowerShell

Exemple 1 : Cet exemple ajoute des balises à la ressource donnée.

```
Add-FSXResourceTag -ResourceARN "arn:aws:fsx:eu-west-1:123456789012:file-system/
fs-01cd23bc4bdf5678a" -Tag @{Key="Users";Value="Test"} -PassThru
```

Sortie :

```
arn:aws:fsx:eu-west-1:123456789012:file-system/fs-01cd23bc4bdf5678a
```

- Pour API plus de détails, consultez la section TagResourceRéférence des AWS Tools for PowerShell applets de commande.

### `Get-FSXBackup`

L'exemple de code suivant montre comment utiliser`Get-FSXBackup`.

Outils pour PowerShell

Exemple 1 : Cet exemple récupère les sauvegardes créées depuis hier pour l'identifiant de système de fichiers donné.

```
Get-FSXBackup -Filter @{Name="file-system-id";Values=$fsx.FileSystemId} | Where-
Object CreationTime -gt (Get-Date).AddDays(-1)
```

Sortie :

```
BackupId        : backup-01dac234e56782bcc
CreationTime    : 6/14/2019 3:35:14 AM
FailureDetails  :
FileSystem      : Amazon.FSx.Model.FileSystem
KmsKeyId        : arn:aws:kms:eu-west-1:123456789012:key/f1af23c4-1b23-1bde-a1f1-
e1234c5af123
Lifecycle       : AVAILABLE
```

```
ProgressPercent : 100
ResourceARN     : arn:aws:fsx:eu-west-1:123456789012:backup/backup-01dac234e56782bcc
Tags            : {}
Type            : AUTOMATIC
```

- Pour API plus de détails, consultez la section DescribeBackupsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-FSXFileSystem`

L'exemple de code suivant montre comment utiliser`Get-FSXFileSystem`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie la description donnéefilesystemId.

```
Get-FSXFileSystem -FileSystemId fs-01cd23bc4bdf5678a
```

Sortie :

```
CreationTime         : 1/17/2019 9:55:30 AM
DNSName              : fs-01cd23bc4bdf5678a.ktmsad.local
FailureDetails       :
FileSystemId         : fs-01cd23bc4bdf5678a
FileSystemType       : WINDOWS
KmsKeyId             : arn:aws:kms:eu-west-1:123456789012:key/f1af23c4-5b67-8bde-
a9f0-e1234c5af678
Lifecycle            : AVAILABLE
LustreConfiguration  :
NetworkInterfaceIds  : {eni-07d1dda1322b7e209}
OwnerId              : 123456789012
ResourceARN          : arn:aws:fsx:eu-west-1:123456789012:file-system/
fs-01cd23bc4bdf5678a
StorageCapacity      : 300
SubnetIds            : {subnet-7d123456}
Tags                 : {FSx-Service}
VpcId                : vpc-41cf2b3f
WindowsConfiguration : Amazon.FSx.Model.WindowsFileSystemConfiguration
```

- Pour API plus de détails, consultez la section DescribeFileSystemsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-FSXResourceTagList`

L'exemple de code suivant montre comment utiliser`Get-FSXResourceTagList`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie les balises pour l'arn de ressource fourni.

```
Get-FSXResourceTagList -ResourceARN $fsx.ResourceARN
```

Sortie :

```
Key         Value
---         -----
FSx-Service Windows
Users       Dev
```

- Pour API plus de détails, consultez la section ListTagsForResourceRéférence des AWS Tools for PowerShell applets de commande.

### `New-FSXBackup`

L'exemple de code suivant montre comment utiliser`New-FSXBackup`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une sauvegarde du système de fichiers donné.

```
New-FSXBackup -FileSystemId fs-0b1fac2345623456ba
```

Sortie :

```
BackupId        : backup-0b1fac2345623456ba
CreationTime    : 6/14/2019 5:37:17 PM
FailureDetails  :
FileSystem      : Amazon.FSx.Model.FileSystem
KmsKeyId        : arn:aws:kms:eu-west-1:123456789012:key/f1af23c4-1b23-1bde-a1f3-
e1234c5af678
Lifecycle       : CREATING
ProgressPercent : 0
```

```
ResourceARN     : arn:aws:fsx:eu-west-1:123456789012:backup/
backup-0b1fac2345623456ba
Tags            : {}
Type            : USER_INITIATED
```

- Pour API plus de détails, consultez la section CreateBackupRéférence des AWS Tools for PowerShell applets de commande.

### `New-FSXFileSystem`

L'exemple de code suivant montre comment utiliser`New-FSXFileSystem`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un nouveau système de fichiers Windows de 300 Go, autorisant l'accès depuis le sous-réseau spécifié, qui prend en charge un débit allant jusqu'à 8 mégaoctets par seconde. Le nouveau système de fichiers est automatiquement joint au Microsoft Active Directory spécifié.

```
New-FSXFileSystem -FileSystemType WINDOWS -StorageCapacity
 300 -SubnetId subnet-1a2b3c4d5e6f -WindowsConfiguration
 @{ThroughputCapacity=8;ActiveDirectoryId='d-1a2b3c4d'}
```

Sortie :

```
CreationTime         : 12/10/2018 6:06:59 PM
DNSName              : fs-abcdef01234567890.example.com
FailureDetails       :
FileSystemId         : fs-abcdef01234567890
FileSystemType       : WINDOWS
KmsKeyId             : arn:aws:kms:us-west-2:123456789012:key/a1234567-252c-45e9-
afaa-123456789abc
Lifecycle            : CREATING
LustreConfiguration  :
NetworkInterfaceIds  : {}
OwnerId              : 123456789012
ResourceARN          : arn:aws:fsx:us-west-2:123456789012:file-system/fs-
abcdef01234567890
StorageCapacity      : 300
SubnetIds            : {subnet-1a2b3c4d5e6f}
Tags                 : {}
VpcId                : vpc-1a2b3c4d5e6f
```

```
WindowsConfiguration : Amazon.FSx.Model.WindowsFileSystemConfiguration
```

- Pour API plus de détails, consultez la section CreateFileSystemRéférence des AWS Tools for PowerShell applets de commande.

### `New-FSXFileSystemFromBackup`

L'exemple de code suivant montre comment utiliser`New-FSXFileSystemFromBackup`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un nouveau système de FSx fichiers Amazon à partir d'une sauvegarde existante du serveur de fichiers Amazon FSx pour Windows.

```
New-FSXFileSystemFromBackup -BackupId $backupID -Tag @{Key="tag:Name";Value="from-
manual-backup"} -SubnetId $SubnetID -SecurityGroupId $SG_ID -WindowsConfiguration
 @{ThroughputCapacity=8;ActiveDirectoryId=$DirectoryID}
```

Sortie :

```
CreationTime         : 8/8/2019 12:59:58 PM
DNSName              : fs-012ff34e56789120.ktmsad.local
FailureDetails       :
FileSystemId         : fs-012ff34e56789120
FileSystemType       : WINDOWS
KmsKeyId             : arn:aws:kms:eu-west-1:123456789012:key/f1af23c4-5b67-1bde-
a2f3-e4567c8a9321
Lifecycle            : CREATING
LustreConfiguration  :
NetworkInterfaceIds  : {}
OwnerId              : 933303704102
ResourceARN          : arn:aws:fsx:eu-west-1:123456789012:file-system/
fs-012ff34e56789120
StorageCapacity      : 300
SubnetIds            : {subnet-fa1ae23c}
Tags                 : {tag:Name}
VpcId                : vpc-12cf3b4f
WindowsConfiguration : Amazon.FSx.Model.WindowsFileSystemConfiguration
```

- Pour API plus de détails, consultez la section CreateFileSystemFromBackupRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-FSXBackup`

L'exemple de code suivant montre comment utiliser`Remove-FSXBackup`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime l'identifiant de sauvegarde indiqué.

```
Remove-FSXBackup -BackupId $backupID
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-FSXBackup (DeleteBackup)" on target
 "backup-0bbca1e2345678e12".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

BackupId                Lifecycle
--------                ---------
backup-0bbca1e2345678e12 DELETED
```

- Pour API plus de détails, consultez la section DeleteBackupRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-FSXFileSystem`

L'exemple de code suivant montre comment utiliser`Remove-FSXFileSystem`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime l'ID de système de FSX fichiers donné.

```
Remove-FSXFileSystem -FileSystemId fs-012ff34e567890120
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-FSXFileSystem (DeleteFileSystem)" on target
 "fs-012ff34e567890120".
```

```
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

FileSystemId         Lifecycle WindowsResponse
------------         --------- ---------------
fs-012ff34e567890120 DELETING  Amazon.FSx.Model.DeleteFileSystemWindowsResponse
```

- Pour API plus de détails, consultez la section DeleteFileSystemRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-FSXResourceTag`

L'exemple de code suivant montre comment utiliser`Remove-FSXResourceTag`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la balise de ressource pour la ressource de système de FSX fichiers donnéeARN.

```
Remove-FSXResourceTag -ResourceARN $FSX.ResourceARN -TagKey Users
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-FSXResourceTag (UntagResource)" on target
 "arn:aws:fsx:eu-west-1:933303704102:file-system/fs-07cd45bc6bdf2674a".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Pour API plus de détails, consultez la section UntagResourceRéférence des AWS Tools for PowerShell applets de commande.

### `Update-FSXFileSystem`

L'exemple de code suivant montre comment utiliser`Update-FSXFileSystem`.

Outils pour PowerShell

Exemple 1 : Cet exemple met à jour les jours de conservation automatique des sauvegardes du système de FSX fichiers via UpdateFileSystemWindowsConfiguration.

```
$UpdateFSXWinConfig = [Amazon.FSx.Model.UpdateFileSystemWindowsConfiguration]::new()
 $UpdateFSXWinConfig.AutomaticBackupRetentionDays = 35
 Update-FSXFileSystem -FileSystemId $FSX.FileSystemId -WindowsConfiguration
 $UpdateFSXWinConfig
```

Sortie :

```
CreationTime         : 1/17/2019 9:55:30 AM
DNSName              : fs-01cd23bc4bdf5678a.ktmsad.local
FailureDetails       :
FileSystemId         : fs-01cd23bc4bdf5678a
FileSystemType       : WINDOWS
KmsKeyId             : arn:aws:kms:eu-west-1:123456789012:key/f1af23c4-1b23-1bde-
a1f2-e1234c5af678
Lifecycle            : AVAILABLE
LustreConfiguration  :
NetworkInterfaceIds  : {eni-01cd23bc4bdf5678a}
OwnerId              : 933303704102
ResourceARN          : arn:aws:fsx:eu-west-1:933303704102:file-system/
fs-07cd45bc6bdf2674a
StorageCapacity      : 300
SubnetIds            : {subnet-1d234567}
Tags                 : {FSx-Service}
VpcId                : vpc-23cf4b5f
WindowsConfiguration : Amazon.FSx.Model.WindowsFileSystemConfiguration
```

- Pour API plus de détails, consultez la section UpdateFileSystemRéférence des AWS Tools for PowerShell applets de commande.

# AWS Glue exemples utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS Tools for PowerShell with AWS Glue.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

# Actions

### `New-GLUEJob`

L'exemple de code suivant montre comment utiliser`New-GLUEJob`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une nouvelle tâche dans AWS Glue. La valeur du nom de commande est toujours**`glueetl`**. AWS Glue permet d'exécuter des scripts de travail écrits en Python ou Scala. Dans cet exemple, le script de tâche (MyTestGlueJob.py) est écrit en Python. Les paramètres Python sont spécifiés dans la **`$DefArgs`** variable, puis transmis à la PowerShell commande dans le **`DefaultArguments`** paramètre, qui accepte une table de hachage. Les paramètres de la **`$JobParams`** variable proviennent de la rubrique CreateJob API Jobs (https://docs.aws.amazon.com/glue/latest/dg/ aws-glue-api-jobs -job.html) de la référence Glue. AWS API

```
$Command = New-Object Amazon.Glue.Model.JobCommand
$Command.Name = 'glueetl'
$Command.ScriptLocation = 's3://amzn-s3-demo-source-bucket/admin/MyTestGlueJob.py'
$Command

$Source = "source_test_table"
$Target = "target_test_table"
$Connections = $Source, $Target

$DefArgs = @{
     '--TempDir' = 's3://amzn-s3-demo-bucket/admin'
     '--job-bookmark-option' = 'job-bookmark-disable'
     '--job-language' = 'python'
     }
$DefArgs

$ExecutionProp = New-Object Amazon.Glue.Model.ExecutionProperty
$ExecutionProp.MaxConcurrentRuns = 1
$ExecutionProp

$JobParams = @{
    "AllocatedCapacity"    = "5"
```

```
    "Command"              = $Command
    "Connections_Connection" = $Connections
    "DefaultArguments"  = $DefArgs
    "Description"       = "This is a test"
    "ExecutionProperty" = $ExecutionProp
    "MaxRetries"        = "1"
    "Name"              = "MyOregonTestGlueJob"
    "Role"              = "Amazon-GlueServiceRoleForSSM"
    "Timeout"           = "20"
     }

New-GlueJob @JobParams
```

- Pour API plus de détails, consultez la section CreateJobRéférence des AWS Tools for PowerShell applets de commande.

# AWS Health exemples utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS Tools for PowerShell with AWS Health.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `Get-HLTHEvent`

L'exemple de code suivant montre comment utiliser`Get-HLTHEvent`.

Exemple 1 : Cette commande renvoie des événements depuis AWS Personal Health Dashboard. L'utilisateur ajoute le paramètre -Region pour voir les événements disponibles pour le service dans la région USA Est (Virginie du Nord), mais le paramètre -Filter_Region filtre les événements enregistrés dans les régions UE (Londres) et USA Ouest (Oregon) (eu-west-2 et us-west-2). Le StartTime paramètre -Filter_ filtre pour une plage d'heures pendant laquelle les événements peuvent commencer, tandis que le EndTime paramètre -Filter_ filtre pour une plage d'heures pendant laquelle les événements peuvent se terminer. Le résultat est un événement de maintenance planifié RDS qui commence dans la plage -Filter_ spécifiée et se termine dans la StartTime plage planifiée EndTime -Filter_.

```
Get-HLTHEvent -Region us-east-1 -Filter_Region "eu-west-2","us-west-2" -
Filter_StartTime @{from="3/14/2019 6:30:00AM";to="3/15/2019 5:00:00PM"} -
Filter_EndTime @{from="3/21/2019 7:00:00AM";to="3/21/2019 5:00:00PM"}
```

Sortie :

```
Arn               : arn:aws:health:us-west-2::event/RDS/
AWS_RDS_HARDWARE_MAINTENANCE_SCHEDULED/
AWS_RDS_HARDWARE_MAINTENANCE_SCHEDULED_USW2_20190314_20190321
AvailabilityZone  :
EndTime           : 3/21/2019 2:00:00 PM
EventTypeCategory : scheduledChange
EventTypeCode     : AWS_RDS_HARDWARE_MAINTENANCE_SCHEDULED
LastUpdatedTime   : 2/28/2019 2:26:07 PM
Region            : us-west-2
Service           : RDS
StartTime         : 3/14/2019 2:00:00 PM
StatusCode        : open
```

- Pour API plus de détails, consultez la section DescribeEventsRéférence des AWS Tools for PowerShell applets de commande.

# IAMexemples utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS Tools for PowerShell withIAM.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques



## Actions

### `Add-IAMClientIDToOpenIDConnectProvider`

L'exemple de code suivant montre comment utiliser`Add-IAMClientIDToOpenIDConnectProvider`.

Outils pour PowerShell

Exemple 1 : Cette commande ajoute l'ID client (ou audience) **`my-application-ID`** au OIDC fournisseur existant nommé**`server.example.com`**.

```
Add-IAMClientIDToOpenIDConnectProvider -ClientID "my-application-ID"
 -OpenIDConnectProviderARN "arn:aws:iam::123456789012:oidc-provider/
server.example.com"
```

- Pour API plus de détails, consultez la section AddClientIdToOpenIdConnectProviderRéférence des AWS Tools for PowerShell applets de commande.

### `Add-IAMRoleTag`

L'exemple de code suivant montre comment utiliser`Add-IAMRoleTag`.

Outils pour PowerShell

Exemple 1 : cet exemple ajoute une balise au rôle dans le service de gestion des identités

```
Add-IAMRoleTag -RoleName AdminRoleacess -Tag @{ Key = 'abac'; Value = 'testing'}
```

- Pour API plus de détails, consultez la section TagRoleRéférence des AWS Tools for PowerShell applets de commande.

### `Add-IAMRoleToInstanceProfile`

L'exemple de code suivant montre comment utiliser`Add-IAMRoleToInstanceProfile`.

Outils pour PowerShell

Exemple 1 : Cette commande ajoute le rôle nommé **`S3Access`** à un profil d'instance existant nommé**`webserver`**. Pour créer le profil d'instance, utilisez la **`New-IAMInstanceProfile`** commande. Après avoir créé le profil d'instance et l'avoir associé à un rôle à l'aide de cette commande, vous pouvez l'associer à une EC2 instance. Pour ce faire, utilisez l'**`New-EC2Instance`**applet de commande avec le paramètre **`InstanceProfile_Arn`** ou le **`InstanceProfile-Name`** paramètre pour lancer la nouvelle instance.

```
Add-IAMRoleToInstanceProfile -RoleName "S3Access" -InstanceProfileName "webserver"
```

- Pour API plus de détails, consultez la section AddRoleToInstanceProfileRéférence des AWS Tools for PowerShell applets de commande.

### `Add-IAMUserTag`

L'exemple de code suivant montre comment utiliser`Add-IAMUserTag`.

Outils pour PowerShell

Exemple 1 : cet exemple ajoute une balise à l'utilisateur dans le service de gestion des identités

```
Add-IAMUserTag -UserName joe -Tag @{ Key = 'abac'; Value = 'testing'}
```

- Pour API plus de détails, consultez la section TagUserRéférence des AWS Tools for PowerShell applets de commande.

### `Add-IAMUserToGroup`

L'exemple de code suivant montre comment utiliser`Add-IAMUserToGroup`.

Outils pour PowerShell

Exemple 1 : Cette commande ajoute l'utilisateur nommé **Bob** au groupe nommé**Admins**.

```
Add-IAMUserToGroup -UserName "Bob" -GroupName "Admins"
```

- Pour API plus de détails, consultez la section AddUserToGroupRéférence des AWS Tools for PowerShell applets de commande.

### `Disable-IAMMFADevice`

L'exemple de code suivant montre comment utiliser`Disable-IAMMFADevice`.

Outils pour PowerShell

Exemple 1 : Cette commande désactive le MFA périphérique matériel associé à l'utilisateur **Bob** qui possède le numéro **123456789012** de série.

```
Disable-IAMMFADevice -UserName "Bob" -SerialNumber "123456789012"
```

Exemple 2 : Cette commande désactive le MFA périphérique virtuel associé à l'utilisateur **David** qui possède le ARN**arn:aws:iam::210987654321:mfa/David**. Notez que le MFA périphérique virtuel n'est pas supprimé du compte. Le périphérique virtuel est toujours présent et apparaît dans la sortie de la **Get-IAMVirtualMFADevice** commande. Avant de créer un nouveau MFA périphérique virtuel pour le même utilisateur, vous devez supprimer l'ancien à l'aide de la **Remove-IAMVirtualMFADevice** commande.

```
Disable-IAMMFADevice -UserName "David" -SerialNumber "arn:aws:iam::210987654321:mfa/
David"
```

- Pour API plus de détails, consultez la section DeactivateMfaDeviceRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-IAMPassword`

L'exemple de code suivant montre comment utiliser`Edit-IAMPassword`.

Outils pour PowerShell

Exemple 1 : Cette commande modifie le mot de passe de l'utilisateur qui exécute la commande. Cette commande ne peut être appelée que par IAM les utilisateurs. Si cette commande est appelée lorsque vous êtes connecté avec les informations d'identification du AWS compte (root), elle renvoie une erreur. **`InvalidUserType`**

```
Edit-IAMPassword -OldPassword "MyOldP@ssw0rd" -NewPassword "MyNewP@ssw0rd"
```

- Pour API plus de détails, consultez la section ChangePasswordRéférence des AWS Tools for PowerShell applets de commande.

### `Enable-IAMMFADevice`

L'exemple de code suivant montre comment utiliser`Enable-IAMMFADevice`.

Outils pour PowerShell

Exemple 1 : Cette commande active le numéro de série du MFA périphérique matériel **`987654321098`** et l'associe à l'utilisateur**`Bob`**. Il inclut les deux premiers codes en séquence provenant de l'appareil.

```
Enable-IAMMFADevice -UserName "Bob" -SerialNumber "987654321098" -
AuthenticationCode1 "12345678" -AuthenticationCode2 "87654321"
```

Exemple 2 : Cet exemple crée et active un MFA périphérique virtuel. La première commande crée le périphérique virtuel et renvoie la représentation de l'objet du périphérique dans la variable**`$MFADevice`**. Vous pouvez utiliser les **`QRCodePng`** propriétés **`.Base32StringSeed`** or pour configurer l'application logicielle de l'utilisateur. La commande finale attribue l'appareil à l'utilisateur**`David`**, en identifiant l'appareil par son numéro de série. La commande synchronise également le dispositif AWS en incluant les deux premiers codes en séquence à partir du MFA dispositif virtuel.

```
$MFADevice = New-IAMVirtualMFADevice -VirtualMFADeviceName "MyMFADevice"
# see example for New-IAMVirtualMFADevice to see how to configure the software
 program with PNG or base32 seed code
Enable-IAMMFADevice -UserName "David" -SerialNumber -SerialNumber
 $MFADevice.SerialNumber -AuthenticationCode1 "24681357" -AuthenticationCode2
 "13572468"
```

- Pour API plus de détails, consultez la section EnableMfaDeviceRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMAccessKey`

L'exemple de code suivant montre comment utiliser`Get-IAMAccessKey`.

Outils pour PowerShell

Exemple 1 : Cette commande répertorie les clés d'accès de l'IAMutilisateur nommé**Bob**. Notez que vous ne pouvez pas répertorier les clés d'accès secrètes des IAM utilisateurs. Si les clés d'accès secrètes sont perdues, vous devez créer de nouvelles clés d'accès avec l'**New-IAMAccessKey**applet de commande.

```
Get-IAMAccessKey -UserName "Bob"
```

Sortie :

```
AccessKeyId              CreateDate                Status              UserName
-----------              ----------                ------              --------
AKIAIOSFODNN7EXAMPLE     12/3/2014 10:53:41 AM     Active              Bob
AKIAI44QH8DHBEXAMPLE     6/6/2013 8:42:26 PM       Inactive            Bob
```

- Pour API plus de détails, consultez la section ListAccessKeysRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMAccessKeyLastUsed`

L'exemple de code suivant montre comment utiliser`Get-IAMAccessKeyLastUsed`.

Outils pour PowerShell

Exemple 1 : renvoie le nom d'utilisateur propriétaire et les informations relatives à la dernière utilisation de la clé d'accès fournie.

```
Get-IAMAccessKeyLastUsed -AccessKeyId ABCDEXAMPLE
```

- Pour API plus de détails, consultez la section GetAccessKeyLastUsedRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMAccountAlias`

L'exemple de code suivant montre comment utiliser`Get-IAMAccountAlias`.

Outils pour PowerShell

Exemple 1 : Cette commande renvoie l'alias de compte pour le Compte AWS.

```
Get-IAMAccountAlias
```

Sortie :

```
ExampleCo
```

- Pour API plus de détails, consultez la section ListAccountAliasesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMAccountAuthorizationDetail`

L'exemple de code suivant montre comment utiliser`Get-IAMAccountAuthorizationDetail`.

Outils pour PowerShell

Exemple 1 : Cet exemple obtient les détails d'autorisation concernant les identités du AWS compte et affiche la liste des éléments de l'objet renvoyé, y compris les utilisateurs, les groupes et les rôles. Par exemple, la **`UserDetailList`** propriété affiche des informations sur les utilisateurs. Des informations similaires sont disponibles dans les **`GroupDetailList`** propriétés **`RoleDetailList`** et.

```
$Details=Get-IAMAccountAuthorizationDetail
$Details
```

Sortie :

```
GroupDetailList : {Administrators, Developers, Testers, Backup}
IsTruncated     : False
Marker          :
RoleDetailList  : {TestRole1, AdminRole, TesterRole, clirole...}
UserDetailList  : {Administrator, Bob, BackupToS3, }
```

```
$Details.UserDetailList
```

Sortie :

```
Arn             : arn:aws:iam::123456789012:user/Administrator
CreateDate      : 10/16/2014 9:03:09 AM
GroupList       : {Administrators}
Path            : /
UserId          : AIDACKCEVSQ6CEXAMPLE1
UserName        : Administrator
UserPolicyList : {}

Arn             : arn:aws:iam::123456789012:user/Bob
CreateDate      : 4/6/2015 12:54:42 PM
GroupList       : {Developers}
Path            : /
UserId          : AIDACKCEVSQ6CEXAMPLE2
UserName        : bab
UserPolicyList : {}

Arn             : arn:aws:iam::123456789012:user/BackupToS3
CreateDate      : 1/27/2015 10:15:08 AM
GroupList       : {Backup}
Path            : /
UserId          : AIDACKCEVSQ6CEXAMPLE3
UserName        : BackupToS3
UserPolicyList : {BackupServicePermissionsToS3Buckets}
```

- Pour API plus de détails, consultez la section GetAccountAuthorizationDetailsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMAccountPasswordPolicy`

L'exemple de code suivant montre comment utiliser`Get-IAMAccountPasswordPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie des informations sur la politique de mot de passe du compte courant. Si aucune politique de mot de passe n'est définie pour le compte, la commande renvoie une **`NoSuchEntity`** erreur.

```
Get-IAMAccountPasswordPolicy
```

Sortie :

```
AllowUsersToChangePassword : True
ExpirePasswords            : True
HardExpiry                 : False
MaxPasswordAge             : 90
MinimumPasswordLength      : 8
PasswordReusePrevention    : 20
RequireLowercaseCharacters : True
RequireNumbers             : True
RequireSymbols             : False
RequireUppercaseCharacters : True
```

- Pour API plus de détails, consultez la section GetAccountPasswordPolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMAccountSummary`

L'exemple de code suivant montre comment utiliser`Get-IAMAccountSummary`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie des informations sur l'utilisation actuelle de l'IAMentité et les quotas d'IAMentité actuels dans le Compte AWS.

```
Get-IAMAccountSummary
```

Sortie :

```
Key                                Value
Users                              7
GroupPolicySizeQuota               5120
PolicyVersionsInUseQuota           10000
ServerCertificatesQuota            20
AccountSigningCertificatesPresent  0
AccountAccessKeysPresent           0
Groups                             3
UsersQuota                         5000
```

```
RolePolicySizeQuota                10240
UserPolicySizeQuota                2048
GroupsPerUserQuota                 10
AssumeRolePolicySizeQuota          2048
AttachedPoliciesPerGroupQuota      2
Roles                              9
VersionsPerPolicyQuota             5
GroupsQuota                        100
PolicySizeQuota                    5120
Policies                           5
RolesQuota                         250
ServerCertificates                 0
AttachedPoliciesPerRoleQuota       2
MFADevicesInUse                    2
PoliciesQuota                      1000
AccountMFAEnabled                  1
Providers                          2
InstanceProfilesQuota              100
MFADevices                         4
AccessKeysPerUserQuota             2
AttachedPoliciesPerUserQuota       2
SigningCertificatesPerUserQuota    2
PolicyVersionsInUse                4
InstanceProfiles                   1
...
```

- Pour API plus de détails, consultez la section GetAccountSummaryRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMAttachedGroupPolicyList`

L'exemple de code suivant montre comment utiliser`Get-IAMAttachedGroupPolicyList`.

Outils pour PowerShell

Exemple 1 : Cette commande renvoie les noms et ARNs les politiques gérées attachés au IAM groupe nommé **`Admins`** dans le AWS compte. Pour voir la liste des politiques intégrées au groupe, utilisez la **`Get-IAMGroupPolicyList`** commande.

```
Get-IAMAttachedGroupPolicyList -GroupName "Admins"
```

Sortie :

```
PolicyArn                                                  PolicyName
---------                                                  ----------
arn:aws:iam::aws:policy/SecurityAudit                      SecurityAudit
arn:aws:iam::aws:policy/AdministratorAccess                AdministratorAccess
```

- Pour API plus de détails, consultez la section ListAttachedGroupPoliciesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMAttachedRolePolicyList`

L'exemple de code suivant montre comment utiliser`Get-IAMAttachedRolePolicyList`.

Outils pour PowerShell

Exemple 1 : Cette commande renvoie les noms et ARNs les politiques gérées associées au IAM rôle nommé **`SecurityAuditRole`** dans le AWS compte. Pour voir la liste des politiques intégrées au rôle, utilisez la **`Get-IAMRolePolicyList`** commande.

```
Get-IAMAttachedRolePolicyList -RoleName "SecurityAuditRole"
```

Sortie :

```
PolicyArn                                                  PolicyName
---------                                                  ----------
arn:aws:iam::aws:policy/SecurityAudit                      SecurityAudit
```

- Pour API plus de détails, consultez la section ListAttachedRolePoliciesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMAttachedUserPolicyList`

L'exemple de code suivant montre comment utiliser`Get-IAMAttachedUserPolicyList`.

Outils pour PowerShell

Exemple 1 : Cette commande renvoie les noms et ARNs les politiques gérées pour l'IAMutilisateur nommé **`Bob`** dans le AWS compte. Pour voir la liste des politiques intégrées intégrées à l'IAMutilisateur, utilisez la **`Get-IAMUserPolicyList`** commande.

```
Get-IAMAttachedUserPolicyList -UserName "Bob"
```

Sortie :

```
PolicyArn                                                 PolicyName
---------                                                 ----------
arn:aws:iam::aws:policy/TesterPolicy                      TesterPolicy
```

- Pour API plus de détails, consultez la section ListAttachedUserPoliciesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMContextKeysForCustomPolicy`

L'exemple de code suivant montre comment utiliser`Get-IAMContextKeysForCustomPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple récupère toutes les clés de contexte présentes dans la politique JSON fournie. Afin de fournir plusieurs politiques, vous pouvez fournir une liste de valeurs séparées par des virgules.

```
$policy1 = '{"Version":"2012-10-17","Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
$policy2 = '{"Version":"2012-10-17","Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/"}}'
Get-IAMContextKeysForCustomPolicy -PolicyInputList $policy1,$policy2
```

- Pour API plus de détails, consultez la section GetContextKeysForCustomPolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMContextKeysForPrincipalPolicy`

L'exemple de code suivant montre comment utiliser`Get-IAMContextKeysForPrincipalPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple récupère toutes les clés de contexte présentes dans le json de politique fourni et les politiques attachées à l'IAMentité (utilisateur/rôle, etc.). Pour : PolicyInputList vous pouvez fournir une liste de valeurs multiples sous forme de valeurs séparées par des virgules.

```
$policy1 = '{"Version":"2012-10-17","Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
$policy2 = '{"Version":"2012-10-17","Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/"}}'
Get-IAMContextKeysForPrincipalPolicy -PolicyInputList $policy1,$policy2 -
PolicySourceArn arn:aws:iam::852640994763:user/TestUser
```

- Pour API plus de détails, consultez la section GetContextKeysForPrincipalPolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMCredentialReport`

L'exemple de code suivant montre comment utiliser`Get-IAMCredentialReport`.

Outils pour PowerShell

Exemple 1 : Cet exemple ouvre le rapport renvoyé et le transmet au pipeline sous la forme d'un tableau de lignes de texte. La première ligne est l'en-tête avec les noms de colonnes séparés par des virgules. Chaque ligne successive est la ligne de détail d'un utilisateur, chaque champ étant séparé par des virgules. Avant de pouvoir consulter le rapport, vous devez le générer à l'aide de l'**`Request-IAMCredentialReport`**applet de commande. Pour récupérer le rapport sous forme de chaîne unique, utilisez à la **`-Raw`** place de**`-AsTextArray`**. L'alias **`-SplitLines`** est également accepté pour le **`-AsTextArray`** commutateur. Pour la liste complète des colonnes de la sortie, consultez la API référence du service. Notez que si vous n'utilisez pas **`-AsTextArray`** ou**`-SplitLines`**, vous devez extraire le texte de la **`.Content`** propriété à l'aide du. NET**`StreamReader`**classe.

```
Request-IAMCredentialReport
```

Sortie :

```
Description                                                          State
-----------                                                          -----
No report exists. Starting a new report generation task              STARTED
```

```
Get-IAMCredentialReport -AsTextArray
```

Sortie :

```
 user,arn,user_creation_time,password_enabled,password_last_used,password_last_changed,passw
 root_account,arn:aws:iam::123456789012:root,2014-10-15T16:31:25+00:00,not_supported,2015-04
A,false,N/A,false,N/A,false,N/A
Administrator,arn:aws:iam::123456789012:user/
Administrator,2014-10-16T16:03:09+00:00,true,2015-04-20T15:18:32+00:00,2014-10-16T16:06:00+0
A,false,true,2014-12-03T18:53:41+00:00,true,2015-03-25T20:38:14+00:00,false,N/
A,false,N/A
Bill,arn:aws:iam::123456789012:user/Bill,2015-04-15T18:27:44+00:00,false,N/A,N/A,N/
A,false,false,N/A,false,N/A,false,2015-04-20T20:00:12+00:00,false,N/A
```

- Pour API plus de détails, consultez la section GetCredentialReportRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMEntitiesForPolicy`

L'exemple de code suivant montre comment utiliser`Get-IAMEntitiesForPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie une liste de IAM groupes, de rôles et d'utilisateurs auxquels la politique **`arn:aws:iam::123456789012:policy/TestPolicy`** est attachée.

```
Get-IAMEntitiesForPolicy -PolicyArn "arn:aws:iam::123456789012:policy/TestPolicy"
```

Sortie :

```
IsTruncated  : False
Marker       :
PolicyGroups : {}
PolicyRoles  : {testRole}
PolicyUsers  : {Bob, Theresa}
```

- Pour API plus de détails, consultez la section ListEntitiesForPolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMGroup`

L'exemple de code suivant montre comment utiliser`Get-IAMGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie des informations sur le IAM groupe**`Testers`**, y compris une collection de tous les IAM utilisateurs appartenant au groupe.

```
$results = Get-IAMGroup -GroupName "Testers"
$results
```

Sortie :

```
Group                                    IsTruncated           Marker
  Users
-----                                    -----------           ------
  -----
Amazon.IdentityManagement.Model.Group    False
  {Theresa, David}
```

```
$results.Group
```

Sortie :

```
Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : 3RHNZZGQJ7QHMAEXAMPLE1
GroupName  : Testers
Path       : /
```

```
$results.Users
```

Sortie :

```
Arn             : arn:aws:iam::123456789012:user/Theresa
```

```
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path             : /
UserId           : 4OSVDDJJTF4XEEXAMPLE2
UserName         : Theresa

Arn              : arn:aws:iam::123456789012:user/David
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path             : /
UserId           : Y4FKWQCXTA52QEXAMPLE3
UserName         : David
```

- Pour API plus de détails, consultez la section GetGroupRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMGroupForUser`

L'exemple de code suivant montre comment utiliser`Get-IAMGroupForUser`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie la liste des IAM groupes auxquels **`David`** appartient l'IAMutilisateur.

```
Get-IAMGroupForUser -UserName David
```

Sortie :

```
Arn        : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId    : 6WCH4TRY3KIHIEXAMPLE1
GroupName  : Administrators
Path       : /

Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : RHNZZGQJ7QHMAEXAMPLE2
GroupName  : Testers
Path       : /

Arn        : arn:aws:iam::123456789012:group/Developers
```

```
CreateDate : 12/10/2014 3:38:55 PM
GroupId    : ZU2EOWMK6WBZOEXAMPLE3
GroupName  : Developers
Path       : /
```

- Pour API plus de détails, consultez la section ListGroupsForUserRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMGroupList`

L'exemple de code suivant montre comment utiliser`Get-IAMGroupList`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie une collection de tous les IAM groupes définis dans le courant Compte AWS.

```
Get-IAMGroupList
```

Sortie :

```
Arn        : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId    : 6WCH4TRY3KIHIEXAMPLE1
GroupName  : Administrators
Path       : /

Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 12/10/2014 3:38:55 PM
GroupId    : ZU2EOWMK6WBZOEXAMPLE2
GroupName  : Developers
Path       : /

Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : RHNZZGQJ7QHMAEXAMPLE3
GroupName  : Testers
Path       : /
```

- Pour API plus de détails, consultez la section ListGroupsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMGroupPolicy`

L'exemple de code suivant montre comment utiliser`Get-IAMGroupPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie des informations sur la politique intégrée nommée **`PowerUserAccess-Testers`** pour le groupe**`Testers`**. La **`PolicyDocument`** propriété est URL codée. Il est décodé dans cet exemple avec le**`UrlDecode`**. NETméthode.

```
$results = Get-IAMGroupPolicy -GroupName Testers -PolicyName PowerUserAccess-Testers
$results
```

Sortie :

```
GroupName     PolicyDocument                                                PolicyName
---------     --------------                                                ----------
Testers       %7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%0A%20...
 PowerUserAccess-Testers

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "NotAction": "iam:*",
      "Resource": "*"
    }
  ]
}
```

- Pour API plus de détails, consultez la section GetGroupPolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMGroupPolicyList`

L'exemple de code suivant montre comment utiliser`Get-IAMGroupPolicyList`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie une liste des politiques intégrées intégrées au groupe**`Testers`**. Pour obtenir les politiques gérées associées au groupe, utilisez la commande**`Get-IAMAttachedGroupPolicyList`**.

```
Get-IAMGroupPolicyList -GroupName Testers
```

Sortie :

```
Deny-Assume-S3-Role-In-Production
PowerUserAccess-Testers
```

- Pour API plus de détails, consultez la section ListGroupPoliciesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMInstanceProfile`

L'exemple de code suivant montre comment utiliser`Get-IAMInstanceProfile`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie les détails du profil d'instance nommé **`ec2instancerole`** défini dans le AWS compte courant.

```
Get-IAMInstanceProfile -InstanceProfileName ec2instancerole
```

Sortie :

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- Pour API plus de détails, consultez la section GetInstanceProfileRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMInstanceProfileForRole`

L'exemple de code suivant montre comment utiliser`Get-IAMInstanceProfileForRole`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie les détails du profil d'instance associé au rôle**`ec2instancerole`**.

```
Get-IAMInstanceProfileForRole -RoleName ec2instancerole
```

Sortie :

```
      Arn                 : arn:aws:iam::123456789012:instance-profile/
ec2instancerole
      CreateDate          : 2/17/2015 2:49:04 PM
      InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
      InstanceProfileName : ec2instancerole
      Path                : /
      Roles               : {ec2instancerole}
```

- Pour API plus de détails, consultez la section ListInstanceProfilesForRoleRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMInstanceProfileList`

L'exemple de code suivant montre comment utiliser`Get-IAMInstanceProfileList`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie une collection des profils d'instance définis dans le fichier actuel Compte AWS.

```
Get-IAMInstanceProfileList
```

Sortie :

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
```

```
Path                : /
Roles               : {ec2instancerole}
```

- Pour API plus de détails, consultez la section ListInstanceProfilesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMLoginProfile`

L'exemple de code suivant montre comment utiliser`Get-IAMLoginProfile`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie la date de création du mot de passe et indique si une réinitialisation du mot de passe est requise pour l'IAMutilisateur**`David`**.

```
Get-IAMLoginProfile -UserName David
```

Sortie :

```
CreateDate                  PasswordResetRequired               UserName
----------                  ---------------------               --------
12/10/2014 3:39:44 PM       False                               David
```

- Pour API plus de détails, consultez la section GetLoginProfileRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMMFADevice`

L'exemple de code suivant montre comment utiliser`Get-IAMMFADevice`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie des informations sur l'MFAappareil attribué à l'IAMutilisateur**`David`**. Dans cet exemple, vous pouvez voir qu'il s'agit d'un périphérique virtuel car il **`SerialNumber`** s'agit d'un numéro de série réel ARN au lieu d'un périphérique physique.

```
Get-IAMMFADevice -UserName David
```

Sortie :

```
EnableDate                SerialNumber                            UserName
----------                ------------                            --------
4/8/2015 9:41:10 AM       arn:aws:iam::123456789012:mfa/David     David
```

- Pour API plus de détails, consultez la section ListMfaDevicesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMOpenIDConnectProvider`

L'exemple de code suivant montre comment utiliser`Get-IAMOpenIDConnectProvider`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie des informations sur le fournisseur OpenID Connect dont ARN le nom est. **`arn:aws:iam::123456789012:oidc-provider/accounts.google.com`** La **`ClientIDList`** propriété est une collection qui contient tous les clients IDs définis pour ce fournisseur.

```
Get-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/oidc.example.com
```

Sortie :

```
ClientIDList         CreateDate               ThumbprintList
        Url
------------         ----------               --------------
        ---
{MyOIDCApp}          2/3/2015 3:00:30 PM
 {12345abcdefghijk67890lmnopqrst98765uvwxy}   oidc.example.com
```

- Pour API plus de détails, consultez la section GetOpenIdConnectProviderRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMOpenIDConnectProviderList`

L'exemple de code suivant montre comment utiliser`Get-IAMOpenIDConnectProviderList`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie une liste ARNS de tous les fournisseurs OpenID Connect définis dans la version actuelle. Compte AWS

```
Get-IAMOpenIDConnectProviderList
```

Sortie :

```
Arn
---
arn:aws:iam::123456789012:oidc-provider/server.example.com
arn:aws:iam::123456789012:oidc-provider/another.provider.com
```

- Pour API plus de détails, consultez la section ListOpenIdConnectProvidersRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMPolicy`

L'exemple de code suivant montre comment utiliser`Get-IAMPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie des informations sur la politique gérée dont le nom ARN est**`arn:aws:iam::123456789012:policy/MySamplePolicy`**.

```
Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

Sortie :

```
Arn              : arn:aws:iam::aws:policy/MySamplePolicy
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : Z27SI6FQMGNQ2EXAMPLE1
PolicyName       : MySamplePolicy
UpdateDate       : 2/6/2015 10:40:08 AM
```

- Pour API plus de détails, consultez la section GetPolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMPolicyList`

L'exemple de code suivant montre comment utiliser`Get-IAMPolicyList`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie une collection des trois premières politiques gérées disponibles dans le AWS compte courant. Comme il n'**`-scope`**est pas spécifié, il utilise par défaut **`all`** et inclut à la fois des politiques AWS gérées et des politiques gérées par le client.

```
Get-IAMPolicyList -MaxItem 3
```

Sortie :

```
Arn              : arn:aws:iam::aws:policy/AWSDirectConnectReadOnlyAccess
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : Z27SI6FQMGNQ2EXAMPLE1
PolicyName       : AWSDirectConnectReadOnlyAccess
UpdateDate       : 2/6/2015 10:40:08 AM

Arn              : arn:aws:iam::aws:policy/AmazonGlacierReadOnlyAccess
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:27 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : NJKMU274MET4EEXAMPLE2
PolicyName       : AmazonGlacierReadOnlyAccess
UpdateDate       : 2/6/2015 10:40:27 AM

Arn              : arn:aws:iam::aws:policy/AWSMarketplaceFullAccess
AttachmentCount  : 0
CreateDate       : 2/11/2015 9:21:45 AM
```

```
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : 5ULJSO2FYVPYGEXAMPLE3
PolicyName       : AWSMarketplaceFullAccess
UpdateDate       : 2/11/2015 9:21:45 AM
```

Exemple 2 : Cet exemple renvoie un ensemble des deux premières politiques gérées par le client disponibles dans le AWS compte courant. Il permet **`-Scope local`** de limiter la sortie aux seules politiques gérées par le client.

```
Get-IAMPolicyList -Scope local -MaxItem 2
```

Sortie :

```
Arn              : arn:aws:iam::123456789012:policy/MyLocalPolicy
AttachmentCount  : 0
CreateDate       : 2/12/2015 9:39:09 AM
DefaultVersionId : v2
Description      :
IsAttachable     : True
Path             : /
PolicyId         : SQVCBLC4VAOUCEXAMPLE4
PolicyName       : MyLocalPolicy
UpdateDate       : 2/12/2015 9:39:53 AM

Arn              : arn:aws:iam::123456789012:policy/policyforec2instancerole
AttachmentCount  : 1
CreateDate       : 2/17/2015 2:51:38 PM
DefaultVersionId : v11
Description      :
IsAttachable     : True
Path             : /
PolicyId         : X5JPBLJH2Z2SOEXAMPLE5
PolicyName       : policyforec2instancerole
UpdateDate       : 2/18/2015 8:52:31 AM
```

- Pour API plus de détails, consultez la section ListPoliciesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMPolicyVersion`

L'exemple de code suivant montre comment utiliser`Get-IAMPolicyVersion`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie le document de politique correspondant à la **v2** version de la politique dont le nom ARN est**arn:aws:iam::123456789012:policy/MyManagedPolicy**. Le document de politique contenu dans la **Document** propriété est URL codé et décodé dans cet exemple avec le**UrlDecode**. NETméthode.

```
$results = Get-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/
MyManagedPolicy -VersionId v2
$results
```

Sortie :

```
CreateDate             Document
 IsDefaultVersion     VersionId
----------             --------
 ----------------     ---------
2/12/2015 9:39:53 AM   %7B%0A%20%20%22Version%22%3A%20%222012-10...    True
        v2

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
$policy = [System.Web.HttpUtility]::UrlDecode($results.Document)
$policy
{
  "Version": "2012-10-17",
  "Statement":
    {
      "Effect": "Allow",
      "Action": "*",
      "Resource": "*"
    }
}
```

- Pour API plus de détails, consultez la section GetPolicyVersionRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMPolicyVersionList`

L'exemple de code suivant montre comment utiliser`Get-IAMPolicyVersionList`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie la liste des versions disponibles de la politique dont ARN c'est le cas**`arn:aws:iam::123456789012:policy/MyManagedPolicy`**. Pour obtenir le document de politique pour une version spécifique, utilisez la **`Get-IAMPolicyVersion`** commande et spécifiez **`VersionId`** celle que vous souhaitez.

```
Get-IAMPolicyVersionList -PolicyArn arn:aws:iam::123456789012:policy/MyManagedPolicy
```

Sortie :

```
CreateDate                    Document                IsDefaultVersion
    VersionId
----------                    --------                ----------------
    ---------
2/12/2015 9:39:53 AM                                  True
    v2
2/12/2015 9:39:09 AM                                  False
    v1
```

- Pour API plus de détails, consultez la section ListPolicyVersionsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMRole`

L'exemple de code suivant montre comment utiliser`Get-IAMRole`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie les détails du**`lamda_exec_role`**. Il inclut le document de politique de confiance qui précise qui peut assumer ce rôle. Le document de politique est URL codé et peut être décodé à l'aide du. NET**`UrlDecode`**méthode. Dans cet exemple, tous les espaces blancs de la politique d'origine avaient été supprimés avant d'être chargés dans la politique. Pour consulter les documents de politique d'autorisation qui déterminent ce que peut faire une personne assumant le rôle, utilisez les documents **`Get-IAMRolePolicy`** pour les politiques intégrées et **`Get-IAMPolicyVersion`** pour les politiques gérées associées.

```
$results = Get-IamRole -RoleName lambda_exec_role
$results | Format-List
```

Sortie :

```
Arn                       : arn:aws:iam::123456789012:role/lambda_exec_role
AssumeRolePolicyDocument : %7B%22Version%22%3A%222012-10-17%22%2C%22Statement%22%3A
%5B%7B%22Sid%22
                           %3A%22%22%2C%22Effect%22%3A%22Allow%22%2C%22Principal
%22%3A%7B%22Service
                           %22%3A%22lambda.amazonaws.com%22%7D%2C%22Action%22%3A
%22sts%3AAssumeRole
                           %22%7D%5D%7D
CreateDate                : 4/2/2015 9:16:11 AM
Path                      : /
RoleId                    : 2YBIKAIBHNKB4EXAMPLE1
RoleName                  : lambda_exec_role
```

```
$policy = [System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
$policy
```

Sortie :

```
{"Version":"2012-10-17","Statement":[{"Sid":"","Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}
```

- Pour API plus de détails, consultez la section GetRoleRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMRoleList`

L'exemple de code suivant montre comment utiliser`Get-IAMRoleList`.

Outils pour PowerShell

Exemple 1 : Cet exemple extrait une liste de tous les IAM rôles du Compte AWS.

```
Get-IAMRoleList
```

Exemple 2 : Cet extrait de code extrait une liste de IAM rôles dans le AWS compte, les affiche trois par trois, puis attend que vous appuyiez sur Entrée entre chaque groupe. Il transmet la **`Marker`** valeur de l'appel précédent pour indiquer où le groupe suivant doit commencer.

```
$nextMarker = $null
Do
{
    $results = Get-IAMRoleList -MaxItem 3 -Marker $nextMarker
    $nextMarker = $AWSHistory.LastServiceResponse.Marker
    $results
    Read-Host
} while ($nextMarker -ne $null)
```

- Pour API plus de détails, consultez la section ListRolesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMRolePolicy`

L'exemple de code suivant montre comment utiliser`Get-IAMRolePolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie le document de politique d'autorisations pour la politique nommée **`oneClick_lambda_exec_role_policy`** qui est intégrée au IAM rôle**`lamda_exec_role`**. Le document de politique qui en résulte est URL codé. Il est décodé dans cet exemple avec le**`UrlDecode`**. NETméthode.

```
$results = Get-IAMRolePolicy -RoleName lambda_exec_role -PolicyName
 oneClick_lambda_exec_role_policy
$results
```

Sortie :

```
PolicyDocument                                          PolicyName
           UserName
--------------                                          ----------
           --------
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...
 oneClick_lambda_exec_role_policy     lambda_exec_role
```

```
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
```

Sortie :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:*"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::*"
      ]
    }
  ]
}
```

- Pour API plus de détails, consultez la section GetRolePolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMRolePolicyList`

L'exemple de code suivant montre comment utiliser`Get-IAMRolePolicyList`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie la liste des noms des politiques intégrées intégrées au IAM rôle**`lamda_exec_role`**. Pour voir les détails d'une politique intégrée, utilisez la commande**`Get-IAMRolePolicy`**.

```
Get-IAMRolePolicyList -RoleName lambda_exec_role
```

Sortie :

```
oneClick_lambda_exec_role_policy
```

- Pour API plus de détails, consultez la section ListRolePoliciesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMRoleTagList`

L'exemple de code suivant montre comment utiliser`Get-IAMRoleTagList`.

Outils pour PowerShell

Exemple 1 : Cet exemple extrait le tag associé au rôle.

```
Get-IAMRoleTagList -RoleName MyRoleName
```

- Pour API plus de détails, consultez la section ListRoleTagsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMSAMLProvider`

L'exemple de code suivant montre comment utiliser`Get-IAMSAMLProvider`.

Outils pour PowerShell

Exemple 1 : Cet exemple récupère les détails du fournisseur SAML 2.0 dont le nom ARM est arn:aws:iam : :123456789012:saml-provider/. SAMLADFS La réponse inclut le document de métadonnées que vous avez obtenu du fournisseur d'identité pour créer l'entité AWS SAML fournisseur ainsi que les dates de création et d'expiration.

```
Get-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS
```

Sortie :

```
CreateDate                SAMLMetadataDocument
     ValidUntil
```

```
----------                  --------------------
     ----------
12/23/2014 12:16:55 PM    <EntityDescriptor ID="_12345678-1234-5678-9012-example1...
    12/23/2114 12:16:54 PM
```

- Pour API plus de détails, consultez la section GetSamlProviderRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMSAMLProviderList`

L'exemple de code suivant montre comment utiliser`Get-IAMSAMLProviderList`.

Outils pour PowerShell

Exemple 1 : Cet exemple récupère la liste des fournisseurs SAML 2.0 créés dans le courant Compte AWS. Il renvoie la ARN date de création et la date d'expiration pour chaque SAML fournisseur.

```
Get-IAMSAMLProviderList
```

Sortie :

```
Arn                                                  CreateDate
 ValidUntil
---                                                  ----------
 ----------
arn:aws:iam::123456789012:saml-provider/SAMLADFS    12/23/2014 12:16:55 PM
 12/23/2114 12:16:54 PM
```

- Pour API plus de détails, reportez-vous à la section L du manuel istSAMLProviders de référence des AWS Tools for PowerShell applets de commande.

### `Get-IAMServerCertificate`

L'exemple de code suivant montre comment utiliser`Get-IAMServerCertificate`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet de récupérer des informations sur le certificat de serveur nommé**`MyServerCertificate`**. Vous trouverez les détails du certificat dans les **`ServerCertificateMetadata`** propriétés **`CertificateBody`** et.

```
$result = Get-IAMServerCertificate -ServerCertificateName MyServerCertificate
$result | format-list
```

Sortie :

```
CertificateBody           : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC

 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6

 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd

 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD

 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                           YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn
+a4GmWIWJ
                           21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/
f0wYK8m9T
                           rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                           nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                           NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                           -----END CERTIFICATE-----
CertificateChain          :
ServerCertificateMetadata :
 Amazon.IdentityManagement.Model.ServerCertificateMetadata
```

```
$result.ServerCertificateMetadata
```

Sortie :

```
Arn                  : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration           : 1/14/2018 9:52:36 AM
Path                 : /Org1/Org2/
ServerCertificateId  : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
UploadDate           : 4/21/2015 11:14:16 AM
```

- Pour API plus de détails, consultez la section GetServerCertificateRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMServerCertificateList`

L'exemple de code suivant montre comment utiliser`Get-IAMServerCertificateList`.

Outils pour PowerShell

Exemple 1 : Cet exemple extrait la liste des certificats de serveur qui ont été téléchargés vers le serveur actuel Compte AWS.

```
Get-IAMServerCertificateList
```

Sortie :

```
Arn                  : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration           : 1/14/2018 9:52:36 AM
Path                 : /Org1/Org2/
ServerCertificateId  : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
UploadDate           : 4/21/2015 11:14:16 AM
```

- Pour API plus de détails, consultez la section ListServerCertificatesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMServiceLastAccessedDetail`

L'exemple de code suivant montre comment utiliser`Get-IAMServiceLastAccessedDetail`.

Outils pour PowerShell

Exemple 1 : Cet exemple fournit les détails du dernier accès au service par l'IAMentité (utilisateur, groupe, rôle ou politique) associée à l'appel de demande.

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

Sortie :

```
f0b7a819-eab0-929b-dc26-ca598911cb9f
```

```
Get-IAMServiceLastAccessedDetail -JobId f0b7a819-eab0-929b-dc26-ca598911cb9f
```

- Pour API plus de détails, consultez la section GetServiceLastAccessedDetailsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMServiceLastAccessedDetailWithEntity`

L'exemple de code suivant montre comment utiliser`Get-IAMServiceLastAccessedDetailWithEntity`.

Outils pour PowerShell

Exemple 1 : Cet exemple fournit l'horodatage du dernier accès au service dans la demande par cette entité respectiveIAM.

```
$results = Get-IAMServiceLastAccessedDetailWithEntity -JobId f0b7a819-eab0-929b-
dc26-ca598911cb9f -ServiceNamespace ec2
$results
```

Sortie :

```
EntityDetailsList : {Amazon.IdentityManagement.Model.EntityDetails}
Error             :
IsTruncated       : False
JobCompletionDate : 12/29/19 11:19:31 AM
JobCreationDate   : 12/29/19 11:19:31 AM
JobStatus         : COMPLETED
Marker            :
```

```
$results.EntityDetailsList
```

Sortie :

```
EntityInfo                                  LastAuthenticated
----------                                  -----------------
Amazon.IdentityManagement.Model.EntityInfo 11/16/19 3:47:00 PM
```

```
$results.EntityInfo
```

Sortie :

```
Arn  : arn:aws:iam::123456789012:user/TestUser
Id   : AIDA4NBK5CXF5TZHU1234
Name : TestUser
Path : /
Type : USER
```

- Pour API plus de détails, consultez la section GetServiceLastAccessedDetailsWithEntitiesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMSigningCertificate`

L'exemple de code suivant montre comment utiliser`Get-IAMSigningCertificate`.

Outils pour PowerShell

Exemple 1 : Cet exemple récupère les détails du certificat de signature associé à l'utilisateur nommé**Bob**.

```
Get-IAMSigningCertificate -UserName Bob
```

Sortie :

```
CertificateBody : -----BEGIN CERTIFICATE-----
                  MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
                  VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
                  b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
```

```
                    BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
                    MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
                    VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
                    b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                    YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
                    21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
                    rDHudUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
                    Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                    nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
                    FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                    NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                    -----END CERTIFICATE-----
CertificateId   : Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
Status          : Active
UploadDate      : 4/20/2015 1:26:01 PM
UserName        : Bob
```

- Pour API plus de détails, consultez la section ListSigningCertificatesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMUser`

L'exemple de code suivant montre comment utiliser`Get-IAMUser`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet de récupérer des informations sur l'utilisateur nommé**`David`**.

```
Get-IAMUser -UserName David
```

Sortie :

```
Arn              : arn:aws:iam::123456789012:user/David
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path             : /
UserId           : Y4FKWQCXTA52QEXAMPLE1
UserName         : David
```

Exemple 2 : Cet exemple permet de récupérer des informations sur l'utilisateur actuellement connecté. IAM

```
Get-IAMUser
```

Sortie :

```
Arn             : arn:aws:iam::123456789012:user/Bob
CreateDate      : 10/16/2014 9:03:09 AM
PasswordLastUsed : 3/4/2015 12:12:33 PM
Path            : /
UserId          : 7K3GJEANSKZF2EXAMPLE2
UserName        : Bob
```

- Pour API plus de détails, consultez la section GetUserRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMUserList`

L'exemple de code suivant montre comment utiliser`Get-IAMUserList`.

Outils pour PowerShell

Exemple 1 : Cet exemple récupère une collection d'utilisateurs dans le fichier actuel Compte AWS.

```
Get-IAMUserList
```

Sortie :

```
Arn             : arn:aws:iam::123456789012:user/Administrator
CreateDate      : 10/16/2014 9:03:09 AM
PasswordLastUsed : 3/4/2015 12:12:33 PM
Path            : /
UserId          : 7K3GJEANSKZF2EXAMPLE1
UserName        : Administrator

Arn             : arn:aws:iam::123456789012:user/Bob
CreateDate      : 4/6/2015 12:54:42 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path            : /
UserId          : L3EWNONDOM3YUEXAMPLE2
UserName        : bab
```

```
    Arn              : arn:aws:iam::123456789012:user/David
    CreateDate       : 12/10/2014 3:39:27 PM
    PasswordLastUsed : 3/19/2015 8:44:04 AM
    Path             : /
    UserId           : Y4FKWQCXTA52QEXAMPLE3
    UserName         : David
```

- Pour API plus de détails, consultez la section ListUsersRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMUserPolicy`

L'exemple de code suivant montre comment utiliser`Get-IAMUserPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple récupère les détails de la politique intégrée nommée **`Davids_IAM_Admin_Policy`** qui est incorporée dans le nom de l'IAMutilisateur. **`David`** Le document de politique est URL codé.

```
$results = Get-IAMUserPolicy -PolicyName Davids_IAM_Admin_Policy -UserName David
$results
```

Sortie :

```
PolicyDocument                                          PolicyName
    UserName
--------------                                          ----------
    --------
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...    Davids_IAM_Admin_Policy
    David

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
```

```
        "iam:*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

- Pour API plus de détails, consultez la section GetUserPolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMUserPolicyList`

L'exemple de code suivant montre comment utiliser`Get-IAMUserPolicyList`.

Outils pour PowerShell

Exemple 1 : Cet exemple extrait la liste des noms des politiques intégrées incorporées dans le nom de l'IAMutilisateur. **`David`**

```
Get-IAMUserPolicyList -UserName David
```

Sortie :

```
Davids_IAM_Admin_Policy
```

- Pour API plus de détails, consultez la section ListUserPoliciesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMUserTagList`

L'exemple de code suivant montre comment utiliser`Get-IAMUserTagList`.

Outils pour PowerShell

Exemple 1 : Cet exemple extrait le tag associé à l'utilisateur.

```
Get-IAMUserTagList -UserName joe
```

- Pour API plus de détails, consultez la section ListUserTagsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-IAMVirtualMFADevice`

L'exemple de code suivant montre comment utiliser`Get-IAMVirtualMFADevice`.

Outils pour PowerShell

Exemple 1 : Cet exemple récupère un ensemble de MFA périphériques virtuels assignés aux utilisateurs dans le AWS compte. La **`User`** propriété de chacun est un objet contenant les détails de l'IAMutilisateur auquel l'appareil est attribué.

```
Get-IAMVirtualMFADevice -AssignmentStatus Assigned
```

Sortie :

```
Base32StringSeed :
EnableDate       : 4/13/2015 12:03:42 PM
QRCodePNG        :
SerialNumber     : arn:aws:iam::123456789012:mfa/David
User             : Amazon.IdentityManagement.Model.User

Base32StringSeed :
EnableDate       : 4/13/2015 12:06:41 PM
QRCodePNG        :
SerialNumber     : arn:aws:iam::123456789012:mfa/root-account-mfa-device
User             : Amazon.IdentityManagement.Model.User
```

- Pour API plus de détails, consultez la section ListVirtualMfaDevicesRéférence des AWS Tools for PowerShell applets de commande.

### `New-IAMAccessKey`

L'exemple de code suivant montre comment utiliser`New-IAMAccessKey`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une nouvelle paire de clés d'accès et de clés d'accès secrètes et les attribue à l'utilisateur**`David`**. Assurez-vous d'enregistrer les **`SecretAccessKey`** valeurs **`AccessKeyId`** et dans un fichier, car c'est la seule fois où vous pouvez obtenir

les**SecretAccessKey**. Vous ne pourrez pas la récupérer ultérieurement. Si vous perdez la clé secrète, vous devez créer une nouvelle paire de clés d'accès.

```
New-IAMAccessKey -UserName David
```

Sortie :

```
AccessKeyId     : AKIAIOSFODNN7EXAMPLE
CreateDate      : 4/13/2015 1:00:42 PM
SecretAccessKey : wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Status          : Active
UserName        : David
```

- Pour API plus de détails, consultez la section CreateAccessKeyRéférence des AWS Tools for PowerShell applets de commande.

### `New-IAMAccountAlias`

L'exemple de code suivant montre comment utiliser`New-IAMAccountAlias`.

Outils pour PowerShell

Exemple 1 : Dans cet exemple, l'alias de votre AWS compte est remplacé par**mycompanyaws**. L'adresse de la page de connexion utilisateur devient panyaws.signin.aws.amazon.com/console. https://mycom L'original URL utilisant votre numéro d'identification de compte au lieu de l'alias (https ://<accountidnumber>.signin.aws.amazon.com/console) continue de fonctionner. Cependant, tout alias défini précédemment URLs cesse de fonctionner.

```
New-IAMAccountAlias -AccountAlias mycompanyaws
```

- Pour API plus de détails, consultez la section CreateAccountAliasRéférence des AWS Tools for PowerShell applets de commande.

### `New-IAMGroup`

L'exemple de code suivant montre comment utiliser`New-IAMGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un nouveau IAM groupe nommé**Developers**.

```
New-IAMGroup -GroupName Developers
```

Sortie :

```
Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 4/14/2015 11:21:31 AM
GroupId    : QNEJ5PM4NFSQCEXAMPLE1
GroupName  : Developers
Path       : /
```

- Pour API plus de détails, consultez la section CreateGroupRéférence des AWS Tools for PowerShell applets de commande.

### `New-IAMInstanceProfile`

L'exemple de code suivant montre comment utiliser`New-IAMInstanceProfile`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un nouveau profil d'IAMinstance nommé**`ProfileForDevEC2Instance`**. Vous devez exécuter la **`Add-IAMRoleToInstanceProfile`** commande séparément pour associer le profil d'instance à un IAM rôle existant qui fournit des autorisations à l'instance. Enfin, attachez le profil d'instance à une EC2 instance lorsque vous la lancez. Pour ce faire, utilisez l'**`New-EC2Instance`**applet de commande avec le paramètre **`InstanceProfile_Arn`** or**`InstanceProfile_Name`**.

```
New-IAMInstanceProfile -InstanceProfileName ProfileForDevEC2Instance
```

Sortie :

```
Arn                 : arn:aws:iam::123456789012:instance-profile/
ProfileForDevEC2Instance
CreateDate          : 4/14/2015 11:31:39 AM
InstanceProfileId   : DYMFXL556EY46EXAMPLE1
InstanceProfileName : ProfileForDevEC2Instance
Path                : /
Roles               : {}
```

- Pour API plus de détails, consultez la section CreateInstanceProfileRéférence des AWS Tools for PowerShell applets de commande.

### `New-IAMLoginProfile`

L'exemple de code suivant montre comment utiliser`New-IAMLoginProfile`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un mot de passe (temporaire) pour l'IAMutilisateur nommé Bob et définit l'indicateur qui oblige l'utilisateur à modifier le mot de passe lors de sa **Bob** prochaine connexion.

```
New-IAMLoginProfile -UserName Bob -Password P@ssw0rd -PasswordResetRequired $true
```

Sortie :

```
CreateDate                  PasswordResetRequired              UserName
----------                  ---------------------              --------
4/14/2015 12:26:30 PM       True                               Bob
```

- Pour API plus de détails, consultez la section CreateLoginProfileRéférence des AWS Tools for PowerShell applets de commande.

### `New-IAMOpenIDConnectProvider`

L'exemple de code suivant montre comment utiliser`New-IAMOpenIDConnectProvider`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un IAM OIDC fournisseur associé au service de fournisseur OIDC compatible trouvé dans URL **`https://example.oidcprovider.com`** et dans l'ID client**`my-testapp-1`**. Le OIDC fournisseur fournit l'empreinte digitale. Pour authentifier l'empreinte numérique, suivez les étapes indiquées sur http://docs.aws.amazon.com/ IAM UserGuide /latest//-thumbprint.html. identity-providers-oidc-obtain

```
New-IAMOpenIDConnectProvider -Url https://example.oidcprovider.com -ClientIDList my-
testapp-1 -ThumbprintList 990F419EXAMPLEECF12DDEDA5EXAMPLE52F20D9E
```

Sortie :

```
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- Pour API plus de détails, consultez la section CreateOpenIdConnectProviderRéférence des AWS Tools for PowerShell applets de commande.

### `New-IAMPolicy`

L'exemple de code suivant montre comment utiliser`New-IAMPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une nouvelle IAM politique dans le AWS compte courant nommée **`MySamplePolicy`** Le fichier **`MySamplePolicy.json`** fournit le contenu de la politique. Notez que vous devez utiliser le paramètre **`-Raw`** switch pour traiter correctement le fichier JSON de régulation.

```
New-IAMPolicy -PolicyName MySamplePolicy -PolicyDocument (Get-Content -Raw
 MySamplePolicy.json)
```

Sortie :

```
Arn              : arn:aws:iam::123456789012:policy/MySamplePolicy
AttachmentCount  : 0
CreateDate       : 4/14/2015 2:45:59 PM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : LD4KP6HVFE7WGEXAMPLE1
PolicyName       : MySamplePolicy
UpdateDate       : 4/14/2015 2:45:59 PM
```

- Pour API plus de détails, consultez la section CreatePolicyRéférence des AWS Tools for PowerShell applets de commande.

### `New-IAMPolicyVersion`

L'exemple de code suivant montre comment utiliser`New-IAMPolicyVersion`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une nouvelle version « v2 » de la IAM politique dont il ARN s'agit **`arn:aws:iam::123456789012:policy/MyPolicy`** et en fait la version par défaut.

Le **NewPolicyVersion.json** fichier fournit le contenu de la politique. Notez que vous devez utiliser le paramètre **-Raw** switch pour traiter correctement le fichier JSON de régulation.

```
New-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -
PolicyDocument (Get-content -Raw NewPolicyVersion.json) -SetAsDefault $true
```

Sortie :

```
CreateDate                          Document                IsDefaultVersion
        VersionId
----------                          --------                ----------------
        ---------
4/15/2015 10:54:54 AM                                       True
        v2
```

- Pour API plus de détails, consultez la section CreatePolicyVersionRéférence des AWS Tools for PowerShell applets de commande.

### New-IAMRole

L'exemple de code suivant montre comment utiliserNew-IAMRole.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un nouveau rôle nommé **MyNewRole** et y attache la politique trouvée dans le fichier**NewRoleTrustPolicy.json**. Notez que vous devez utiliser le paramètre **-Raw** switch pour traiter correctement le fichier JSON de régulation. Le document de politique affiché dans la sortie est URL codé. Il est décodé dans cet exemple avec le**UrlDecode**. NETméthode.

```
$results = New-IAMRole -AssumeRolePolicyDocument (Get-Content -raw
 NewRoleTrustPolicy.json) -RoleName MyNewRole
$results
```

Sortie :

```
Arn                       : arn:aws:iam::123456789012:role/MyNewRole
AssumeRolePolicyDocument : %7B%0D%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%0D
%0A%20%20%22Statement%22
```

```
                            %3A%20%5B%0D%0A%20%20%20%20%7B%0D%0A
%20%20%20%20%20%20%22Sid%22%3A%20%22%22%2C
                            %0D%0A%20%20%20%20%20%20%22Effect%22%3A%20%22Allow%22%2C
%0D%0A%20%20%20%20%20%20
                            %22Principal%22%3A%20%7B%0D%0A
%20%20%20%20%20%20%20%20%22AWS%22%3A%20%22arn%3Aaws
                            %3Aiam%3A%3A123456789012%3ADavid%22%0D%0A
%20%20%20%20%20%20%7D%2C%0D%0A%20%20%20
                            %20%20%20%22Action%22%3A%20%22sts%3AAssumeRole%22%0D%0A
%20%20%20%20%7D%0D%0A%20
                            %20%5D%0D%0A%7D
CreateDate               : 4/15/2015 11:04:23 AM
Path                     : /
RoleId                   : V5PAJI2KPN4EAEXAMPLE1
RoleName                 : MyNewRole

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:David"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- Pour API plus de détails, consultez la section CreateRoleRéférence des AWS Tools for PowerShell applets de commande.

### `New-IAMSAMLProvider`

L'exemple de code suivant montre comment utiliser`New-IAMSAMLProvider`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une nouvelle entité SAML fournisseur dansIAM. Il est nommé **`MySAMLProvider`** et décrit par le document de SAML métadonnées trouvé dans le

fichier**SAMLMetaData.xml**, qui a été téléchargé séparément depuis le site Web du fournisseur de SAML services.

```
New-IAMSAMLProvider -Name MySAMLProvider -SAMLMetadataDocument (Get-Content -Raw
 SAMLMetaData.xml)
```

Sortie :

```
arn:aws:iam::123456789012:saml-provider/MySAMLProvider
```

- Pour API plus de détails, reportez-vous à la section C reateSAMLProvider dans le manuel de référence des AWS Tools for PowerShell applets de commande.

### `New-IAMServiceLinkedRole`

L'exemple de code suivant montre comment utiliser`New-IAMServiceLinkedRole`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un rôle lié à un service pour le service de mise à l'échelle automatique.

```
New-IAMServiceLinkedRole -AWSServiceName autoscaling.amazonaws.com -CustomSuffix
 RoleNameEndsWithThis -Description "My service-linked role to support autoscaling"
```

- Pour API plus de détails, consultez la section CreateServiceLinkedRoleRéférence des AWS Tools for PowerShell applets de commande.

### `New-IAMUser`

L'exemple de code suivant montre comment utiliser`New-IAMUser`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un IAM utilisateur nommé**Bob**. Si Bob doit se connecter à la AWS console, vous devez exécuter la commande séparément **`New-IAMLoginProfile`** pour créer un profil de connexion avec un mot de passe. Si Bob doit exécuter AWS PowerShell des CLI commandes multiplateformes ou AWS API passer des appels, vous devez exécuter la **`New-IAMAccessKey`** commande séparément pour créer des clés d'accès.

```
New-IAMUser -UserName Bob
```

Sortie :

```
Arn              : arn:aws:iam::123456789012:user/Bob
CreateDate       : 4/22/2015 12:02:11 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path             : /
UserId           : AIDAJWGEFDMEMEXAMPLE1
UserName         : Bob
```

- Pour API plus de détails, consultez la section CreateUserRéférence des AWS Tools for PowerShell applets de commande.

### `New-IAMVirtualMFADevice`

L'exemple de code suivant montre comment utiliser`New-IAMVirtualMFADevice`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un nouveau MFA périphérique virtuel. Les lignes 2 et 3 extraient la **`Base32StringSeed`** valeur dont le MFA logiciel virtuel a besoin pour créer un compte (comme alternative au code QR). Après avoir configuré le programme avec la valeur, obtenez deux codes d'authentification séquentiels auprès du programme. Enfin, utilisez la dernière commande pour lier le MFA périphérique virtuel à l'IAMutilisateur **`Bob`** et synchroniser le compte avec les deux codes d'authentification.

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$SR = New-Object System.IO.StreamReader($Device.Base32StringSeed)
$base32stringseed = $SR.ReadToEnd()
$base32stringseed
CZWZMCQNW4DEXAMPLE3VOUGXJFZYSUW7EXAMPLECR4NJFD65GX2SLUDW2EXAMPLE
```

Sortie :

```
-- Pause here to enter base-32 string seed code into virtual MFA program to register
 account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

Exemple 2 : Cet exemple crée un nouveau MFA périphérique virtuel. Les lignes 2 et 3 extraient la **QRCodePNG** valeur et l'écrivaient dans un fichier. Cette image peut être numérisée par le MFA logiciel virtuel pour créer un compte (au lieu de saisir manuellement la StringSeed valeur Base32). Après avoir créé le compte dans votre MFA programme virtuel, obtenez deux codes d'authentification séquentiels et saisissez-les dans les dernières commandes pour relier le MFA périphérique virtuel à l'IAMutilisateur **Bob** et synchroniser le compte.

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$BR = New-Object System.IO.BinaryReader($Device.QRCodePNG)
$BR.ReadBytes($BR.BaseStream.Length) | Set-Content -Encoding Byte -Path QRCode.png
```

Sortie :

```
 -- Pause here to scan PNG with virtual MFA program to register account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

- Pour API plus de détails, consultez la section CreateVirtualMfaDeviceRéférence des AWS Tools for PowerShell applets de commande.

### `Publish-IAMServerCertificate`

L'exemple de code suivant montre comment utiliser`Publish-IAMServerCertificate`.

Outils pour PowerShell

Exemple 1 : Cet exemple télécharge un nouveau certificat de serveur sur le IAM compte. Les fichiers contenant le corps du certificat, la clé privée et (éventuellement) la chaîne de certificats doivent tous être PEM codés. Notez que les paramètres nécessitent le contenu réel des fichiers plutôt que les noms de fichiers. Vous devez utiliser le paramètre **-Raw** switch pour traiter correctement le contenu du fichier.

```
Publish-IAMServerCertificate -ServerCertificateName MyTestCert -CertificateBody
 (Get-Content -Raw server.crt) -PrivateKey (Get-Content -Raw server.key)
```

Sortie :

```
Arn                 : arn:aws:iam::123456789012:server-certificate/MyTestCert
```

```
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /
ServerCertificateId   : ASCAJIEXAMPLE7J7HQZYW
ServerCertificateName : MyTestCert
UploadDate            : 4/21/2015 11:14:16 AM
```

- Pour API plus de détails, consultez la section UploadServerCertificateRéférence des AWS Tools for PowerShell applets de commande.

### `Publish-IAMSigningCertificate`

L'exemple de code suivant montre comment utiliser`Publish-IAMSigningCertificate`.

Outils pour PowerShell

Exemple 1 : Cet exemple télécharge un nouveau certificat de signature X.509 et l'associe à l'IAMutilisateur nommé. **`Bob`** Le fichier contenant le corps du certificat est PEM codé. Le **`CertificateBody`** paramètre nécessite le contenu réel du fichier de certificat plutôt que le nom du fichier. Vous devez utiliser le paramètre **`-Raw`** switch pour traiter correctement le fichier.

```
Publish-IAMSigningCertificate -UserName Bob -CertificateBody (Get-Content -Raw
 SampleSigningCert.pem)
```

Sortie :

```
CertificateBody : -----BEGIN CERTIFICATE-----
                  MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
                  VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
                  b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
                  BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
                  MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
                  VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
                  b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                  YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
                  21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
                  rDHudUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
                  Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                  nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
                  FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                  NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                  -----END CERTIFICATE-----
```

```
CertificateId   : Y3EK7RMEXAMPLESV33FCEXAMPLEHMJLU
Status          : Active
UploadDate      : 4/20/2015 1:26:01 PM
UserName        : Bob
```

- Pour API plus de détails, consultez la section UploadSigningCertificateRéférence des AWS Tools for PowerShell applets de commande.

### `Register-IAMGroupPolicy`

L'exemple de code suivant montre comment utiliser`Register-IAMGroupPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple associe la politique gérée par le client nommée **`TesterPolicy`** au IAM groupe**`Testers`**. Les utilisateurs de ce groupe sont immédiatement affectés par les autorisations définies dans la version par défaut de cette politique.

```
Register-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterPolicy
```

Exemple 2 : Cet exemple associe la politique AWS gérée nommée **`AdministratorAccess`** au IAM groupe**`Admins`**. Les utilisateurs de ce groupe sont immédiatement affectés par les autorisations définies dans la dernière version de cette politique.

```
Register-IAMGroupPolicy -GroupName Admins -PolicyArn arn:aws:iam::aws:policy/
AdministratorAccess
```

- Pour API plus de détails, consultez la section AttachGroupPolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Register-IAMRolePolicy`

L'exemple de code suivant montre comment utiliser`Register-IAMRolePolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple associe la politique AWS gérée nommée **`SecurityAudit`** au IAM rôle**`CoSecurityAuditors`**. Les utilisateurs qui assument ce rôle sont immédiatement affectés par les autorisations définies dans la dernière version de cette politique.

```
Register-IAMRolePolicy -RoleName CoSecurityAuditors -PolicyArn
 arn:aws:iam::aws:policy/SecurityAudit
```

- Pour API plus de détails, consultez la section AttachRolePolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Register-IAMUserPolicy`

L'exemple de code suivant montre comment utiliser`Register-IAMUserPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple attache la politique AWS gérée nommée **`AmazonCognitoPowerUser`** à l'IAMutilisateur**`Bob`**. L'utilisateur est immédiatement affecté par les autorisations définies dans la dernière version de cette politique.

```
Register-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::aws:policy/
AmazonCognitoPowerUser
```

- Pour API plus de détails, consultez la section AttachUserPolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-IAMAccessKey`

L'exemple de code suivant montre comment utiliser`Remove-IAMAccessKey`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la paire de clés AWS d'accès avec l'ID **`AKIAIOSFODNN7EXAMPLE`** de clé de l'utilisateur nommé**`Bob`**.

```
Remove-IAMAccessKey -AccessKeyId AKIAIOSFODNN7EXAMPLE -UserName Bob -Force
```

- Pour API plus de détails, consultez la section DeleteAccessKeyRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-IAMAccountAlias`

L'exemple de code suivant montre comment utiliser`Remove-IAMAccountAlias`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime l'alias de compte de votre Compte AWS. La page de connexion de l'utilisateur avec l'alias https://mycom panyaws.signin.aws.amazon.com/console ne fonctionne plus. Vous devez plutôt utiliser l'original URL avec votre numéro d' Compte AWS identification sur https ://.signin.aws.amazon.com/console. <accountidnumber>

```
Remove-IAMAccountAlias -AccountAlias mycompanyaws
```

- Pour API plus de détails, consultez la section DeleteAccountAliasRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-IAMAccountPasswordPolicy`

L'exemple de code suivant montre comment utiliser`Remove-IAMAccountPasswordPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la politique de mot de passe pour le Compte AWS et rétablit toutes les valeurs par défaut d'origine. Si aucune politique de mot de passe n'existe actuellement, le message d'erreur suivant s'affiche : PasswordPolicy Impossible de trouver la politique de compte portant le nom.

```
Remove-IAMAccountPasswordPolicy
```

- Pour API plus de détails, consultez la section DeleteAccountPasswordPolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-IAMClientIDFromOpenIDConnectProvider`

L'exemple de code suivant montre comment utiliser`Remove-IAMClientIDFromOpenIDConnectProvider`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime l'ID client **`My-TestApp-3`** de la liste des clients IDs associés au IAM OIDC fournisseur dont l'identifiant ARN est**`arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com`**.

```
Remove-IAMClientIDFromOpenIDConnectProvider -ClientID My-TestApp-3
 -OpenIDConnectProviderArn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com
```

- Pour API plus de détails, consultez la section RemoveClientIdFromOpenIdConnectProviderRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-IAMGroup`

L'exemple de code suivant montre comment utiliser`Remove-IAMGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le IAM groupe nommé**`MyTestGroup`**. La première commande supprime tous IAM les utilisateurs membres du groupe, tandis que la seconde supprime le IAM groupe. Les deux commandes fonctionnent sans qu'aucune demande de confirmation ne soit requise.

```
(Get-IAMGroup -GroupName MyTestGroup).Users | Remove-IAMUserFromGroup -GroupName
 MyTestGroup -Force
Remove-IAMGroup -GroupName MyTestGroup -Force
```

- Pour API plus de détails, consultez la section DeleteGroupRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-IAMGroupPolicy`

L'exemple de code suivant montre comment utiliser`Remove-IAMGroupPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la politique en ligne nommée **`TesterPolicy`** du IAM groupe**`Testers`**. Les utilisateurs de ce groupe perdent immédiatement les autorisations définies dans cette politique.

```
Remove-IAMGroupPolicy -GroupName Testers -PolicyName TestPolicy
```

- Pour API plus de détails, consultez la section DeleteGroupPolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-IAMInstanceProfile`

L'exemple de code suivant montre comment utiliser`Remove-IAMInstanceProfile`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le profil d'EC2instance nommé**`MyAppInstanceProfile`**. La première commande détache tous les rôles du profil d'instance, puis la deuxième commande supprime le profil d'instance.

```
(Get-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile).Roles | Remove-
IAMRoleFromInstanceProfile -InstanceProfileName MyAppInstanceProfile
Remove-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile
```

- Pour API plus de détails, consultez la section DeleteInstanceProfileRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-IAMLoginProfile`

L'exemple de code suivant montre comment utiliser`Remove-IAMLoginProfile`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le profil de connexion de l'IAMutilisateur nommé**`Bob`**. Cela empêche l'utilisateur de se connecter à la AWS console. Cela n'empêche pas l'utilisateur d'exécuter des AWS CLI API appels à l'aide de clés AWS d'accès qui pourraient encore être associées au compte utilisateur. PowerShell

```
Remove-IAMLoginProfile -UserName Bob
```

- Pour API plus de détails, consultez la section DeleteLoginProfileRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-IAMOpenIDConnectProvider`

L'exemple de code suivant montre comment utiliser`Remove-IAMOpenIDConnectProvider`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le IAM OIDC fournisseur qui se connecte au fournisseur**`example.oidcprovider.com`**. Assurez-vous de mettre à jour ou de supprimer tous les rôles qui font référence à ce fournisseur dans l'**`Principal`**élément de la politique de confiance du rôle.

```
Remove-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- Pour API plus de détails, consultez la section DeleteOpenIdConnectProviderRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-IAMPolicy`

L'exemple de code suivant montre comment utiliser`Remove-IAMPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la politique dont le nom ARN est**`arn:aws:iam::123456789012:policy/MySamplePolicy`**. Avant de pouvoir supprimer la politique, vous devez d'abord supprimer toutes les versions, à l'exception de la version par défaut, en exécutant la commande**`Remove-IAMPolicyVersion`**. Vous devez également dissocier la politique de tous IAM les utilisateurs, groupes ou rôles.

```
Remove-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

Exemple 2 : cet exemple supprime une politique en supprimant d'abord toutes les versions de stratégie autres que celles par défaut, en la détachant de toutes les IAM entités attachées, puis en supprimant la politique elle-même. La première ligne permet de récupérer l'objet de politique. La deuxième ligne récupère toutes les versions de politique qui ne sont pas signalées comme version par défaut dans une collection, puis supprime chaque politique de la collection. La troisième ligne récupère tous les IAM utilisateurs, groupes et rôles auxquels la politique est attachée. Les lignes quatre à six détachent la politique de chaque entité attachée. La dernière ligne utilise cette commande pour supprimer la politique gérée ainsi que la version par défaut restante. L'exemple inclut le paramètre **`-Force`** switch sur toute ligne qui en a besoin pour supprimer les demandes de confirmation.

```
$pol = Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

```
Get-IAMPolicyVersions -PolicyArn $pol.Arn | where {-not $_.IsDefaultVersion} |
 Remove-IAMPolicyVersion -PolicyArn $pol.Arn -force
$attached = Get-IAMEntitiesForPolicy -PolicyArn $pol.Arn
$attached.PolicyGroups | Unregister-IAMGroupPolicy -PolicyArn $pol.arn
$attached.PolicyRoles | Unregister-IAMRolePolicy -PolicyArn $pol.arn
$attached.PolicyUsers | Unregister-IAMUserPolicy -PolicyArn $pol.arn
Remove-IAMPolicy $pol.Arn -Force
```

- Pour API plus de détails, consultez la section DeletePolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-IAMPolicyVersion`

L'exemple de code suivant montre comment utiliser`Remove-IAMPolicyVersion`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la version identifiée comme étant dans **v2** la politique dont le nom ARN est**arn:aws:iam::123456789012:policy/MySamplePolicy**.

```
Remove-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy -
VersionID v2
```

Exemple 2 : Cet exemple supprime une politique en supprimant d'abord toutes les versions de stratégie autres que celles par défaut, puis en supprimant la politique elle-même. La première ligne permet de récupérer l'objet de politique. La deuxième ligne récupère toutes les versions de politique qui ne sont pas signalées comme étant par défaut dans une collection, puis utilise cette commande pour supprimer chaque politique de la collection. La dernière ligne supprime la politique elle-même ainsi que la version par défaut restante. Notez que pour supprimer correctement une politique gérée, vous devez également la détacher de tous les utilisateurs, groupes ou rôles à l'aide des **Unregister-IAMRolePolicy** commandes **Unregister-IAMUserPolicyUnregister-IAMGroupPolicy**, et. Consultez l'exemple de l'**Remove-IAMPolicy**applet de commande.

```
$pol = Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
Get-IAMPolicyVersions -PolicyArn $pol.Arn | where {-not $_.IsDefaultVersion} |
 Remove-IAMPolicyVersion -PolicyArn $pol.Arn -force
Remove-IAMPolicy -PolicyArn $pol.Arn -force
```

- Pour API plus de détails, consultez la section DeletePolicyVersionRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-IAMRole`

L'exemple de code suivant montre comment utiliser`Remove-IAMRole`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le rôle nommé **`MyNewRole`** du IAM compte courant. Avant de pouvoir supprimer le rôle, vous devez d'abord utiliser la **`Unregister-IAMRolePolicy`** commande pour détacher les politiques gérées. Les politiques intégrées sont supprimées avec le rôle.

```
Remove-IAMRole -RoleName MyNewRole
```

Exemple 2 : Cet exemple détache toutes les politiques gérées du rôle nommé, **`MyNewRole`** puis supprime le rôle. La première ligne récupère toutes les politiques gérées associées au rôle en tant que collection, puis détache chaque politique de la collection du rôle. La deuxième ligne supprime le rôle lui-même. Les politiques intégrées sont supprimées en même temps que le rôle.

```
Get-IAMAttachedRolePolicyList -RoleName MyNewRole | Unregister-IAMRolePolicy -
RoleName MyNewRole
Remove-IAMRole -RoleName MyNewRole
```

- Pour API plus de détails, consultez la section DeleteRoleRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-IAMRoleFromInstanceProfile`

L'exemple de code suivant montre comment utiliser`Remove-IAMRoleFromInstanceProfile`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le rôle nommé **`MyNewRole`** du profil d'EC2instance nommé**`MyNewRole`**. Un profil d'instance créé dans la IAM console porte toujours le même nom que le rôle, comme dans cet exemple. Si vous les créez dans le API ouCLI, ils peuvent porter des noms différents.

```
Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyNewRole -RoleName MyNewRole
 -Force
```

- Pour API plus de détails, consultez la section RemoveRoleFromInstanceProfileRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-IAMRolePermissionsBoundary`

L'exemple de code suivant montre comment utiliser`Remove-IAMRolePermissionsBoundary`.

Outils pour PowerShell

Exemple 1 : Cet exemple montre comment supprimer la limite d'autorisation attachée à un IAM rôle.

```
Remove-IAMRolePermissionsBoundary -RoleName MyRoleName
```

- Pour API plus de détails, consultez la section DeleteRolePermissionsBoundaryRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-IAMRolePolicy`

L'exemple de code suivant montre comment utiliser`Remove-IAMRolePolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la politique **`S3AccessPolicy`** intégrée intégrée au IAM rôle. **`S3BackupRole`**

```
Remove-IAMRolePolicy -PolicyName S3AccessPolicy -RoleName S3BackupRole
```

- Pour API plus de détails, consultez la section DeleteRolePolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-IAMRoleTag`

L'exemple de code suivant montre comment utiliser`Remove-IAMRoleTag`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la balise du rôle nommé « MyRoleName » avec la clé de balise « abac ». Pour supprimer plusieurs balises, fournissez une liste de clés de balises séparées par des virgules.

```
Remove-IAMRoleTag -RoleName MyRoleName -TagKey "abac","xyzw"
```

- Pour API plus de détails, consultez la section UntagRoleRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-IAMSAMLProvider`

L'exemple de code suivant montre comment utiliser`Remove-IAMSAMLProvider`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le fournisseur IAM SAML 2.0 dont le nom ARN est**`arn:aws:iam::123456789012:saml-provider/SAMLADFSProvider`**.

```
Remove-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFSProvider
```

- Pour API plus de détails, reportez-vous à la section D eleteSAMLProvider dans la référence des AWS Tools for PowerShell applets de commande.

### `Remove-IAMServerCertificate`

L'exemple de code suivant montre comment utiliser`Remove-IAMServerCertificate`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le certificat de serveur nommé**`MyServerCert`**.

```
Remove-IAMServerCertificate -ServerCertificateName MyServerCert
```

- Pour API plus de détails, consultez la section DeleteServerCertificateRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-IAMServiceLinkedRole`

L'exemple de code suivant montre comment utiliser`Remove-IAMServiceLinkedRole`.

Outils pour PowerShell

Exemple 1 : Cet exemple a supprimé le rôle lié au service. Notez que si le service utilise toujours ce rôle, cette commande entraîne un échec.

```
Remove-IAMServiceLinkedRole -RoleName
 AWSServiceRoleForAutoScaling_RoleNameEndsWithThis
```

- Pour API plus de détails, consultez la section DeleteServiceLinkedRoleRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-IAMSigningCertificate`

L'exemple de code suivant montre comment utiliser`Remove-IAMSigningCertificate`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le certificat de signature avec l'ID **`Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU`** de l'IAMutilisateur nommé**`Bob`**.

```
Remove-IAMSigningCertificate -UserName Bob -CertificateId
 Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
```

- Pour API plus de détails, consultez la section DeleteSigningCertificateRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-IAMUser`

L'exemple de code suivant montre comment utiliser`Remove-IAMUser`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime l'IAMutilisateur nommé**`Bob`**.

```
Remove-IAMUser -UserName Bob
```

Exemple 2 : Cet exemple supprime l'IAMutilisateur nommé **Theresa** ainsi que tous les éléments qui doivent être supprimés en premier.

```
$name = "Theresa"

# find any groups and remove user from them
$groups = Get-IAMGroupForUser -UserName $name
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName -
UserName $name -Force }

# find any inline policies and delete them
$inlinepols = Get-IAMUserPolicies -UserName $name
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
 $name -Force}

# find any managed polices and detach them
$managedpols = Get-IAMAttachedUserPolicies -UserName $name
foreach ($pol in $managedpols) { Unregister-IAMUserPolicy -PolicyArn $pol.PolicyArn
 -UserName $name }

# find any signing certificates and delete them
$certs = Get-IAMSigningCertificate -UserName $name
foreach ($cert in $certs) { Remove-IAMSigningCertificate -CertificateId
 $cert.CertificateId -UserName $name -Force }

# find any access keys and delete them
$keys = Get-IAMAccessKey -UserName $name
foreach ($key in $keys) { Remove-IAMAccessKey -AccessKeyId $key.AccessKeyId -
UserName $name -Force }

# delete the user's login profile, if one exists - note: need to use try/catch to
 suppress not found error
try { $prof = Get-IAMLoginProfile -UserName $name -ea 0 } catch { out-null }
if ($prof) { Remove-IAMLoginProfile -UserName $name -Force }

# find any MFA device, detach it, and if virtual, delete it.
$mfa = Get-IAMMFADevice -UserName $name
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -SerialNumber
 $mfa.SerialNumber }
}
```

```
# finally, remove the user
Remove-IAMUser -UserName $name -Force
```

- Pour API plus de détails, consultez la section DeleteUserRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-IAMUserFromGroup`

L'exemple de code suivant montre comment utiliser`Remove-IAMUserFromGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime l'IAMutilisateur **`Bob`** du groupe**`Testers`**.

```
Remove-IAMUserFromGroup -GroupName Testers -UserName Bob
```

Exemple 2 : Cet exemple permet de rechercher tous les groupes dont un IAM utilisateur **`Theresa`** est membre, puis **`Theresa`** de les supprimer de ces groupes.

```
$groups = Get-IAMGroupForUser -UserName Theresa
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName -
UserName Theresa -Force }
```

Exemple 3 : Cet exemple montre une autre méthode pour supprimer l'IAMutilisateur **`Bob`** du **`Testers`** groupe.

```
Get-IAMGroupForUser -UserName Bob | Remove-IAMUserFromGroup -UserName Bob -GroupName
 Testers -Force
```

- Pour API plus de détails, consultez la section RemoveUserFromGroupRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-IAMUserPermissionsBoundary`

L'exemple de code suivant montre comment utiliser`Remove-IAMUserPermissionsBoundary`.

Outils pour PowerShell

Exemple 1 : Cet exemple montre comment supprimer la limite d'autorisation attachée à un IAM utilisateur.

```
Remove-IAMUserPermissionsBoundary -UserName joe
```

- Pour API plus de détails, consultez la section DeleteUserPermissionsBoundaryRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-IAMUserPolicy`

L'exemple de code suivant montre comment utiliser`Remove-IAMUserPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la politique intégrée nommée **`AccessToEC2Policy`** qui est incorporée dans le nom de l'IAMutilisateur. **`Bob`**

```
Remove-IAMUserPolicy -PolicyName AccessToEC2Policy -UserName Bob
```

Exemple 2 : Cet exemple recherche toutes les politiques intégrées qui sont intégrées dans le nom de l'IAMutilisateur, **`Theresa`** puis les supprime.

```
$inlinepols = Get-IAMUserPolicies -UserName Theresa
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
 Theresa -Force}
```

- Pour API plus de détails, consultez la section DeleteUserPolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-IAMUserTag`

L'exemple de code suivant montre comment utiliser`Remove-IAMUserTag`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le tag de l'utilisateur nommé « joe » avec les clés de tag « abac » et « xyzw ». Pour supprimer plusieurs balises, fournissez une liste de clés de balises séparées par des virgules.

```
Remove-IAMUserTag -UserName joe -TagKey "abac","xyzw"
```

- Pour API plus de détails, consultez la section UntagUserRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-IAMVirtualMFADevice`

L'exemple de code suivant montre comment utiliser`Remove-IAMVirtualMFADevice`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le MFA périphérique IAM virtuel dont le nom ARN est**`arn:aws:iam::123456789012:mfa/bob`**.

```
Remove-IAMVirtualMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/bob
```

Exemple 2 : Cet exemple vérifie si un MFA appareil est attribué à l'IAMutilisateur Theresa. S'il en trouve un, l'appareil est désactivé pour l'IAMutilisateur. Si le périphérique est virtuel, il est également supprimé.

```
$mfa = Get-IAMMFADevice -UserName Theresa
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -SerialNumber
 $mfa.SerialNumber }
}
```

- Pour API plus de détails, consultez la section DeleteVirtualMfaDeviceRéférence des AWS Tools for PowerShell applets de commande.

### `Request-IAMCredentialReport`

L'exemple de code suivant montre comment utiliser`Request-IAMCredentialReport`.

Outils pour PowerShell

Exemple 1 : Cet exemple demande la génération d'un nouveau rapport, qui peut être effectué toutes les quatre heures. Si le dernier rapport est encore récent, le champ État est libellé comme suit**`COMPLETE`**. **`Get-IAMCredentialReport`**À utiliser pour afficher le rapport complet.

```
Request-IAMCredentialReport
```

Sortie :

```
Description                                                    State
-----------                                                    -----
No report exists. Starting a new report generation task        STARTED
```

- Pour API plus de détails, consultez la section GenerateCredentialReportRéférence des AWS Tools for PowerShell applets de commande.

### `Request-IAMServiceLastAccessedDetail`

L'exemple de code suivant montre comment utiliser`Request-IAMServiceLastAccessedDetail`.

Outils pour PowerShell

Exemple 1 : Cet exemple est une applet de commande équivalente à. GenerateServiceLastAccessedDetails API Cela fournit un identifiant de tâche qui peut être utilisé dans IAMServiceLastAccessedDetail Get- et Get- IAMServiceLastAccessedDetailWithEntity

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

- Pour API plus de détails, consultez la section GenerateServiceLastAccessedDetailsRéférence des AWS Tools for PowerShell applets de commande.

### `Set-IAMDefaultPolicyVersion`

L'exemple de code suivant montre comment utiliser`Set-IAMDefaultPolicyVersion`.

Outils pour PowerShell

Exemple 1 : Cet exemple définit la **v2** version de la politique qui ARN est **arn:aws:iam::123456789012:policy/MyPolicy** la version active par défaut.

```
Set-IAMDefaultPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -
VersionId v2
```

- Pour API plus de détails, consultez la section SetDefaultPolicyVersionRéférence des AWS Tools for PowerShell applets de commande.

### `Set-IAMRolePermissionsBoundary`

L'exemple de code suivant montre comment utiliser`Set-IAMRolePermissionsBoundary`.

Outils pour PowerShell

Exemple 1 : Cet exemple montre comment définir la limite d'autorisation pour un IAM rôle. Vous pouvez définir des politiques AWS gérées ou des politiques personnalisées comme limite d'autorisation.

```
Set-IAMRolePermissionsBoundary -RoleName MyRoleName -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- Pour API plus de détails, consultez la section PutRolePermissionsBoundaryRéférence des AWS Tools for PowerShell applets de commande.

### `Set-IAMUserPermissionsBoundary`

L'exemple de code suivant montre comment utiliser`Set-IAMUserPermissionsBoundary`.

Outils pour PowerShell

Exemple 1 : Cet exemple montre comment définir la limite d'autorisation pour l'utilisateur. Vous pouvez définir des politiques AWS gérées ou des politiques personnalisées comme limite d'autorisation.

```
Set-IAMUserPermissionsBoundary -UserName joe -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- Pour API plus de détails, consultez la section PutUserPermissionsBoundaryRéférence des AWS Tools for PowerShell applets de commande.

### `Sync-IAMMFADevice`

L'exemple de code suivant montre comment utiliser`Sync-IAMMFADevice`.

Outils pour PowerShell

Exemple 1 : Cet exemple synchronise le MFA périphérique associé à l'IAMutilisateur **`arn:aws:iam::123456789012:mfa/bob`** avec un programme **`Bob`** d'authentification qui a fourni les deux codes d'authentification. ARN

```
Sync-IAMMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/theresa -
AuthenticationCode1 123456 -AuthenticationCode2 987654 -UserName Bob
```

Exemple 2 : Cet exemple synchronise le IAM MFA périphérique associé à l'IAMutilisateur **`Theresa`** avec un appareil physique qui possède le numéro de série **`ABCD12345678`** et qui a fourni les deux codes d'authentification.

```
Sync-IAMMFADevice -SerialNumber ABCD12345678 -AuthenticationCode1 123456 -
AuthenticationCode2 987654 -UserName Theresa
```

- Pour API plus de détails, consultez la section ResyncMfaDeviceRéférence des AWS Tools for PowerShell applets de commande.

### `Unregister-IAMGroupPolicy`

L'exemple de code suivant montre comment utiliser`Unregister-IAMGroupPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple détache la politique de groupe géré qui ARN **`arn:aws:iam::123456789012:policy/TesterAccessPolicy`** provient du groupe nommé**`Testers`**.

```
Unregister-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterAccessPolicy
```

Exemple 2 : Cet exemple trouve toutes les politiques gérées associées au groupe nommé **`Testers`** et les détache du groupe.

```
Get-IAMAttachedGroupPolicies -GroupName Testers | Unregister-IAMGroupPolicy -
Groupname Testers
```

- Pour API plus de détails, consultez la section DetachGroupPolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Unregister-IAMRolePolicy`

L'exemple de code suivant montre comment utiliser`Unregister-IAMRolePolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple détache la politique de groupe géré qui ARN est liée **arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy** au rôle nommé**FedTesterRole**.

```
Unregister-IAMRolePolicy -RoleName FedTesterRole -PolicyArn
 arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy
```

Exemple 2 : Cet exemple recherche toutes les politiques gérées associées au rôle nommé **FedTesterRole** et les détache du rôle.

```
Get-IAMAttachedRolePolicyList -RoleName FedTesterRole | Unregister-IAMRolePolicy -
Rolename FedTesterRole
```

- Pour API plus de détails, consultez la section DetachRolePolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Unregister-IAMUserPolicy`

L'exemple de code suivant montre comment utiliser`Unregister-IAMUserPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple détache la politique gérée qui ARN **arn:aws:iam::123456789012:policy/TesterPolicy** provient de l'IAMutilisateur nommé**Bob**.

```
Unregister-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::123456789012:policy/
TesterPolicy
```

Exemple 2 : Cet exemple trouve toutes les politiques gérées associées à l'IAMutilisateur nommé **Theresa** et détache ces politiques de l'utilisateur.

```
Get-IAMAttachedUserPolicyList -UserName Theresa | Unregister-IAMUserPolicy -Username
 Theresa
```

- Pour API plus de détails, consultez la section DetachUserPolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Update-IAMAccessKey`

L'exemple de code suivant montre comment utiliser`Update-IAMAccessKey`.

Outils pour PowerShell

Exemple 1 : Cet exemple modifie le statut de la clé d'accès **`AKIAIOSFODNN7EXAMPLE`** pour l'IAMutilisateur nommé **`Bob`** to**`Inactive`**.

```
Update-IAMAccessKey -UserName Bob -AccessKeyId AKIAIOSFODNN7EXAMPLE -Status Inactive
```

- Pour API plus de détails, consultez la section UpdateAccessKeyRéférence des AWS Tools for PowerShell applets de commande.

### `Update-IAMAccountPasswordPolicy`

L'exemple de code suivant montre comment utiliser`Update-IAMAccountPasswordPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple met à jour la politique de mot de passe du compte avec les paramètres spécifiés. Notez que les paramètres qui ne sont pas inclus dans la commande ne sont pas laissés inchangés. Au lieu de cela, ils sont réinitialisés aux valeurs par défaut.

```
Update-IAMAccountPasswordPolicy -AllowUsersToChangePasswords $true -HardExpiry
 $false -MaxPasswordAge 90 -MinimumPasswordLength 8 -PasswordReusePrevention 20
 -RequireLowercaseCharacters $true -RequireNumbers $true -RequireSymbols $true -
RequireUppercaseCharacters $true
```

- Pour API plus de détails, consultez la section UpdateAccountPasswordPolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Update-IAMAssumeRolePolicy`

L'exemple de code suivant montre comment utiliser`Update-IAMAssumeRolePolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple met à jour le IAM rôle nommé **`ClientRole`** avec une nouvelle politique de confiance dont le contenu provient du fichier**`ClientRolePolicy.json`**. Notez que vous devez utiliser le paramètre **`-Raw`** switch pour traiter correctement le contenu du JSON fichier.

```
Update-IAMAssumeRolePolicy -RoleName ClientRole -PolicyDocument (Get-Content -raw
 ClientRolePolicy.json)
```

- Pour API plus de détails, consultez la section UpdateAssumeRolePolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Update-IAMGroup`

L'exemple de code suivant montre comment utiliser`Update-IAMGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple renomme le IAM groupe **`Testers`** en**`AppTesters`**.

```
Update-IAMGroup -GroupName Testers -NewGroupName AppTesters
```

Exemple 2 : Cet exemple change le chemin du IAM groupe **`AppTesters`** en**`/Org1/Org2/`**. Cela change le ARN pour le groupe en**`arn:aws:iam::123456789012:group/Org1/Org2/AppTesters`**.

```
Update-IAMGroup -GroupName AppTesters -NewPath /Org1/Org2/
```

- Pour API plus de détails, consultez la section UpdateGroupRéférence des AWS Tools for PowerShell applets de commande.

### `Update-IAMLoginProfile`

L'exemple de code suivant montre comment utiliser`Update-IAMLoginProfile`.

Outils pour PowerShell

Exemple 1 : Cet exemple définit un nouveau mot de passe temporaire pour IAM l'**`Bob`**utilisateur et demande à ce dernier de le modifier lors de sa prochaine connexion.

```
Update-IAMLoginProfile -UserName Bob -Password "P@ssw0rd1234" -PasswordResetRequired
 $true
```

- Pour API plus de détails, consultez la section UpdateLoginProfileRéférence des AWS Tools for PowerShell applets de commande.

### `Update-IAMOpenIDConnectProviderThumbprint`

L'exemple de code suivant montre comment utiliser`Update-IAMOpenIDConnectProviderThumbprint`.

Outils pour PowerShell

Exemple 1 : Cet exemple met à jour la liste des empreintes numériques des certificats pour le OIDC fournisseur qui doit **`arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com`** utiliser ARN une nouvelle empreinte numérique. Le OIDC fournisseur partage la nouvelle valeur lorsque le certificat associé au fournisseur change.

```
Update-IAMOpenIDConnectProviderThumbprint -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com -ThumbprintList
 7359755EXAMPLEabc3060bce3EXAMPLEec4542a3
```

- Pour API plus de détails, consultez la section UpdateOpenIdConnectProviderThumbprintRéférence des AWS Tools for PowerShell applets de commande.

### `Update-IAMRole`

L'exemple de code suivant montre comment utiliser`Update-IAMRole`.

Outils pour PowerShell

Exemple 1 : Cet exemple met à jour la description du rôle et la durée maximale de session (en secondes) pour laquelle la session d'un rôle peut être demandée.

```
Update-IAMRole -RoleName MyRoleName -Description "My testing role" -
MaxSessionDuration 43200
```

- Pour API plus de détails, consultez la section UpdateRoleRéférence des AWS Tools for PowerShell applets de commande.

### `Update-IAMRoleDescription`

L'exemple de code suivant montre comment utiliser`Update-IAMRoleDescription`.

Outils pour PowerShell

Exemple 1 : Cet exemple met à jour la description d'un IAM rôle dans votre compte.

```
Update-IAMRoleDescription -RoleName MyRoleName -Description "My testing role"
```

- Pour API plus de détails, consultez la section UpdateRoleDescriptionRéférence des AWS Tools for PowerShell applets de commande.

### `Update-IAMSAMLProvider`

L'exemple de code suivant montre comment utiliser`Update-IAMSAMLProvider`.

Outils pour PowerShell

Exemple 1 : Cet exemple met à jour le SAML fournisseur IAM dans lequel ARN se trouve **`arn:aws:iam::123456789012:saml-provider/SAMLADFS`** un nouveau document de SAML métadonnées issu du fichier**`SAMLMetaData.xml`**. Notez que vous devez utiliser le paramètre **`-Raw`** switch pour traiter correctement le contenu du JSON fichier.

```
Update-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS -SAMLMetadataDocument (Get-Content -Raw SAMLMetaData.xml)
```

- Pour API plus de détails, consultez la section UpdateSamlProviderRéférence des AWS Tools for PowerShell applets de commande.

### `Update-IAMServerCertificate`

L'exemple de code suivant montre comment utiliser`Update-IAMServerCertificate`.

Outils pour PowerShell

Exemple 1 : Cet exemple renomme le certificat nommé **`MyServerCertificate`** en**`MyRenamedServerCertificate`**.

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -
NewServerCertificateName MyRenamedServerCertificate
```

Exemple 2 : Cet exemple déplace le certificat nommé **`MyServerCertificate`** vers le chemin /Org1/Org2/. Cela change le ARN pour la ressource

en**arn:aws:iam::123456789012:server-certificate/Org1/Org2/MyServerCertificate**.

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -NewPath /
Org1/Org2/
```

- Pour API plus de détails, consultez la section UpdateServerCertificateRéférence des AWS Tools for PowerShell applets de commande.

### `Update-IAMSigningCertificate`

L'exemple de code suivant montre comment utiliser`Update-IAMSigningCertificate`.

Outils pour PowerShell

Exemple 1 : Cet exemple met à jour le certificat associé à l'IAMutilisateur nommé **Bob** et dont l'ID de certificat est **Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU** destiné à le marquer comme inactif.

```
Update-IAMSigningCertificate -CertificateId Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU -
UserName Bob -Status Inactive
```

- Pour API plus de détails, consultez la section UpdateSigningCertificateRéférence des AWS Tools for PowerShell applets de commande.

### `Update-IAMUser`

L'exemple de code suivant montre comment utiliser`Update-IAMUser`.

Outils pour PowerShell

Exemple 1 : Cet exemple renomme l'IAMutilisateur **Bob** en**Robert**.

```
Update-IAMUser -UserName Bob -NewUserName Robert
```

Exemple 2 : Cet exemple modifie le chemin de l'IAMutilisateur **Bob** vers**/Org1/Org2/**, ce qui change effectivement le chemin ARN de l'utilisateur vers**arn:aws:iam::123456789012:user/Org1/Org2/bob**.

```
Update-IAMUser -UserName Bob -NewPath /Org1/Org2/
```

- Pour API plus de détails, consultez la section UpdateUserRéférence des AWS Tools for PowerShell applets de commande.

### `Write-IAMGroupPolicy`

L'exemple de code suivant montre comment utiliser`Write-IAMGroupPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une politique intégrée nommée **`AppTesterPolicy`** et l'intègre dans le IAM groupe. **`AppTesters`** Si une politique intégrée portant le même nom existe déjà, elle est remplacée. Le contenu JSON de la politique est inclus dans le fichier**`apptesterpolicy.json`**. Notez que vous devez utiliser le **`-Raw`** paramètre pour traiter correctement le contenu du JSON fichier.

```
Write-IAMGroupPolicy -GroupName AppTesters -PolicyName AppTesterPolicy -
PolicyDocument (Get-Content -Raw apptesterpolicy.json)
```

- Pour API plus de détails, consultez la section PutGroupPolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Write-IAMRolePolicy`

L'exemple de code suivant montre comment utiliser`Write-IAMRolePolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une politique intégrée nommée **`FedTesterRolePolicy`** et l'intègre dans le IAM rôle. **`FedTesterRole`** Si une politique intégrée portant le même nom existe déjà, elle est remplacée. Le contenu JSON de la politique provient du fichier**`FedTesterPolicy.json`**. Notez que vous devez utiliser le **`-Raw`** paramètre pour traiter correctement le contenu du JSON fichier.

```
Write-IAMRolePolicy -RoleName FedTesterRole -PolicyName FedTesterRolePolicy -
PolicyDocument (Get-Content -Raw FedTesterPolicy.json)
```

- Pour API plus de détails, consultez la section PutRolePolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Write-IAMUserPolicy`

L'exemple de code suivant montre comment utiliser`Write-IAMUserPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une politique intégrée nommée **`EC2AccessPolicy`** et l'intègre à l'IAMutilisateur. **`Bob`** Si une politique intégrée portant le même nom existe déjà, elle est remplacée. Le contenu JSON de la politique provient du fichier**`EC2AccessPolicy.json`**. Notez que vous devez utiliser le **`-Raw`** paramètre pour traiter correctement le contenu du JSON fichier.

```
Write-IAMUserPolicy -UserName Bob -PolicyName EC2AccessPolicy -PolicyDocument (Get-
Content -Raw EC2AccessPolicy.json)
```

- Pour API plus de détails, consultez la section PutUserPolicyRéférence des AWS Tools for PowerShell applets de commande.

# Exemples de Kinesis utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS Tools for PowerShell aide de Winesis.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `Get-KINRecord`

L'exemple de code suivant montre comment utiliser`Get-KINRecord`.

Exemple 1 : Cet exemple montre comment renvoyer et extraire des données d'une série d'un ou de plusieurs enregistrements. L'itérateur fourni à Get- KINRecord détermine la position de départ des enregistrements à renvoyer qui, dans cet exemple, sont capturés dans une variable, $records. Chaque enregistrement individuel est ensuite accessible en indexant la collection $records. En supposant que les données de l'enregistrement sont du texte codé en UTF -8, la commande finale indique comment extraire les données de l' MemoryStream objet et les renvoyer sous forme de texte à la console.

```
$records
$records = Get-KINRecord -ShardIterator "AAAAAAAAAAGIc....9VnbiRNaP"
```

Sortie :

```
MillisBehindLatest NextShardIterator            Records
------------------ -----------------            -------
0                  AAAAAAAAAAERNIq...uDn11HuUs  {Key1, Key2}
```

```
$records.Records[0]
```

Sortie :

```
ApproximateArrivalTimestamp Data                   PartitionKey SequenceNumber
--------------------------- ----                   ------------ --------------
3/7/2016 5:14:33 PM         System.IO.MemoryStream Key1
 4955986459776...931586
```

```
[Text.Encoding]::UTF8.GetString($records.Records[0].Data.ToArray())
```

Sortie :

```
test data from string
```

- Pour API plus de détails, consultez la section GetRecordsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-KINShardIterator`

L'exemple de code suivant montre comment utiliser`Get-KINShardIterator`.

Outils pour PowerShell

Exemple 1 : renvoie un itérateur de partition pour la partition et la position de départ spécifiées. Les détails des identificateurs de partition et des numéros de séquence peuvent être obtenus à partir de la sortie de l'KINStreamapplet de commande Get-, en faisant référence à la collection Shards de l'objet de flux renvoyé. L'itérateur renvoyé peut être utilisé avec l'KINRecordapplet de commande Get- pour extraire des enregistrements de données dans la partition.

```
Get-KINShardIterator -StreamName "mystream" -ShardId "shardId-000000000000" -
ShardIteratorType AT_SEQUENCE_NUMBER -StartingSequenceNumber "495598645..."
```

Sortie :

```
AAAAAAAAAAGIc....9VnbiRNaP
```

- Pour API plus de détails, consultez la section GetShardIteratorRéférence des AWS Tools for PowerShell applets de commande.

### `Get-KINStream`

L'exemple de code suivant montre comment utiliser`Get-KINStream`.

Outils pour PowerShell

Exemple 1 : renvoie les détails du flux spécifié.

```
Get-KINStream -StreamName "mystream"
```

Sortie :

```
HasMoreShards        : False
RetentionPeriodHours : 24
Shards               : {}
StreamARN            : arn:aws:kinesis:us-west-2:123456789012:stream/mystream
StreamName           : mystream
StreamStatus         : ACTIVE
```

- Pour API plus de détails, consultez la section DescribeStreamRéférence des AWS Tools for PowerShell applets de commande.

### `New-KINStream`

L'exemple de code suivant montre comment utiliser`New-KINStream`.

Outils pour PowerShell

Exemple 1 : crée un nouveau flux. Par défaut, cette applet de commande ne renvoie aucune sortie. Le PassThru commutateur - est donc ajouté pour renvoyer la valeur fournie au StreamName paramètre - pour une utilisation ultérieure.

```
$streamName = New-KINStream -StreamName "mystream" -ShardCount 1 -PassThru
```

- Pour API plus de détails, consultez la section CreateStreamRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-KINStream`

L'exemple de code suivant montre comment utiliser`Remove-KINStream`.

Outils pour PowerShell

Exemple 1 : Supprime le flux spécifié. Vous êtes invité à confirmer avant l'exécution de la commande. Pour supprimer les demandes de confirmation, utilisez le commutateur -Force.

```
Remove-KINStream -StreamName "mystream"
```

- Pour API plus de détails, consultez la section DeleteStreamRéférence des AWS Tools for PowerShell applets de commande.

### `Write-KINRecord`

L'exemple de code suivant montre comment utiliser`Write-KINRecord`.

Outils pour PowerShell

Exemple 1 : écrit un enregistrement contenant la chaîne fournie au paramètre -Text.

```
Write-KINRecord -Text "test data from string" -StreamName "mystream" -PartitionKey
 "Key1"
```

Exemple 2 : écrit un enregistrement contenant les données contenues dans le fichier spécifié. Le fichier est traité comme une séquence d'octets. S'il contient du texte, il doit être écrit avec le codage nécessaire avant de l'utiliser avec cette applet de commande.

```
Write-KINRecord -FilePath "C:\TestData.txt" -StreamName "mystream" -PartitionKey
 "Key2"
```

- Pour API plus de détails, consultez la section PutRecordRéférence des AWS Tools for PowerShell applets de commande.

# Exemples Lambda utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS Tools for PowerShell aide de Lambda.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `Add-LMResourceTag`

L'exemple de code suivant montre comment utiliser`Add-LMResourceTag`.

Outils pour PowerShell

Exemple 1 : ajoute les trois balises (Washington, Oregon et Californie) et leurs valeurs associées à la fonction spécifiée identifiée par sonARN.

```
Add-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction" -Tag @{ "Washington" = "Olympia"; "Oregon"
 = "Salem"; "California" = "Sacramento" }
```

- Pour API plus de détails, consultez la section TagResourceRéférence des AWS Tools for PowerShell applets de commande.

### `Get-LMAccountSetting`

L'exemple de code suivant montre comment utiliser`Get-LMAccountSetting`.

Outils pour PowerShell

Exemple 1 : Cet exemple s'affiche pour comparer la limite du compte et l'utilisation du compte

```
Get-LMAccountSetting | Select-Object
 @{Name="TotalCodeSizeLimit";Expression={$_.AccountLimit.TotalCodeSize}},
 @{Name="TotalCodeSizeUsed";Expression={$_.AccountUsage.TotalCodeSize}}
```

Sortie :

```
TotalCodeSizeLimit TotalCodeSizeUsed
------------------ -----------------
       80530636800          15078795
```

- Pour API plus de détails, consultez la section GetAccountSettingsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-LMAlias`

L'exemple de code suivant montre comment utiliser`Get-LMAlias`.

Outils pour PowerShell

Exemple 1 : Cet exemple récupère les poids de la configuration de routage pour un alias de fonction Lambda spécifique.

```
Get-LMAlias -FunctionName "MylambdaFunction123" -Name "newlabel1" -Select
 RoutingConfig
```

Sortie :

```
AdditionalVersionWeights
------------------------
{[1, 0.6]}
```

- Pour API plus de détails, consultez la section GetAliasRéférence des AWS Tools for PowerShell applets de commande.

### `Get-LMFunctionConcurrency`

L'exemple de code suivant montre comment utiliser`Get-LMFunctionConcurrency`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir la simultanéité réservée pour la fonction Lambda

```
Get-LMFunctionConcurrency -FunctionName "MylambdaFunction123" -Select *
```

Sortie :

```
ReservedConcurrentExecutions
----------------------------
100
```

- Pour API plus de détails, consultez la section GetFunctionConcurrencyRéférence des AWS Tools for PowerShell applets de commande.

### `Get-LMFunctionConfiguration`

L'exemple de code suivant montre comment utiliser`Get-LMFunctionConfiguration`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie la configuration spécifique à la version d'une fonction Lambda.

```
Get-LMFunctionConfiguration -FunctionName "MylambdaFunction123" -Qualifier
 "PowershellAlias"
```

Sortie :

```
CodeSha256                  : uWOW0R7z+f0VyLuUg7+/D08hkMFsq0SF4seuyUZJ/R8=
CodeSize                    : 1426
DeadLetterConfig            : Amazon.Lambda.Model.DeadLetterConfig
Description                 : Verson 3 to test Aliases
Environment                 : Amazon.Lambda.Model.EnvironmentResponse
FunctionArn                 : arn:aws:lambda:us-
east-1:123456789012:function:MylambdaFunction123
                              :PowershellAlias
FunctionName                : MylambdaFunction123
Handler                     : lambda_function.launch_instance
KMSKeyArn                   :
LastModified                : 2019-12-25T09:52:59.872+0000
LastUpdateStatus            : Successful
LastUpdateStatusReason      :
LastUpdateStatusReasonCode  :
Layers                      : {}
MasterArn                   :
MemorySize                  : 128
RevisionId                  : 5d7de38b-87f2-4260-8f8a-e87280e10c33
Role                        : arn:aws:iam::123456789012:role/service-role/lambda
Runtime                     : python3.8
State                       : Active
StateReason                 :
StateReasonCode             :
Timeout                     : 600
TracingConfig               : Amazon.Lambda.Model.TracingConfigResponse
Version                     : 4
VpcConfig                   : Amazon.Lambda.Model.VpcConfigDetail
```

- Pour API plus de détails, consultez la section GetFunctionConfigurationRéférence des AWS Tools for PowerShell applets de commande.

### `Get-LMFunctionList`

L'exemple de code suivant montre comment utiliser`Get-LMFunctionList`.

Outils pour PowerShell

Exemple 1 : Cet exemple affiche toutes les fonctions Lambda avec une taille de code triée

```
Get-LMFunctionList | Sort-Object -Property CodeSize | Select-Object FunctionName,
 RunTime, Timeout, CodeSize
```

Sortie :

```
FunctionName                                                   Runtime   Timeout
 CodeSize
------------                                                   -------   -------
 --------
test                                                           python2.7       3
 243
MylambdaFunction123                                            python3.8     600
 659
myfuncpython1                                                  python3.8     303
 675
```

- Pour API plus de détails, consultez la section ListFunctionsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-LMPolicy`

L'exemple de code suivant montre comment utiliser`Get-LMPolicy`.

Outils pour PowerShell

Exemple 1 : Cet exemple affiche la politique de fonction de la fonction Lambda

```
Get-LMPolicy -FunctionName test -Select Policy
```

Sortie :

```
{"Version":"2012-10-17","Id":"default","Statement":
[{"Sid":"xxxx","Effect":"Allow","Principal":
{"Service":"sns.amazonaws.com"},"Action":"lambda:InvokeFunction","Resource":"arn:aws:lambda:
east-1:123456789102:function:test"}]}
```

- Pour API plus de détails, consultez la section GetPolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Get-LMProvisionedConcurrencyConfig`

L'exemple de code suivant montre comment utiliser`Get-LMProvisionedConcurrencyConfig`.

Outils pour PowerShell

Exemple 1 : Cet exemple obtient la configuration de simultanéité provisionnée pour l'alias spécifié de la fonction Lambda.

```
C:\>Get-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -
Qualifier "NewAlias1"
```

Sortie :

```
AllocatedProvisionedConcurrentExecutions : 0
AvailableProvisionedConcurrentExecutions : 0
LastModified                             : 2020-01-15T03:21:26+0000
RequestedProvisionedConcurrentExecutions : 70
Status                                   : IN_PROGRESS
StatusReason                             :
```

- Pour API plus de détails, consultez la section GetProvisionedConcurrencyConfigRéférence des AWS Tools for PowerShell applets de commande.

### `Get-LMProvisionedConcurrencyConfigList`

L'exemple de code suivant montre comment utiliser`Get-LMProvisionedConcurrencyConfigList`.

Outils pour PowerShell

Exemple 1 : Cet exemple extrait la liste des configurations de simultanéité provisionnées pour une fonction Lambda.

```
Get-LMProvisionedConcurrencyConfigList -FunctionName "MylambdaFunction123"
```

- Pour API plus de détails, consultez la section ListProvisionedConcurrencyConfigsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-LMResourceTag`

L'exemple de code suivant montre comment utiliser`Get-LMResourceTag`.

Outils pour PowerShell

Exemple 1 : récupère les balises et leurs valeurs actuellement définies sur la fonction spécifiée.

```
Get-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction"
```

Sortie :

```
Key        Value
---        -----
California Sacramento
Oregon     Salem
Washington Olympia
```

- Pour API plus de détails, consultez la section ListTagsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-LMVersionsByFunction`

L'exemple de code suivant montre comment utiliser`Get-LMVersionsByFunction`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie la liste des configurations spécifiques à chaque version de la fonction Lambda.

```
Get-LMVersionsByFunction -FunctionName "MylambdaFunction123"
```

Sortie :

```
FunctionName        Runtime   MemorySize Timeout CodeSize LastModified
   RoleName
------------        -------   ---------- ------- -------- ------------
   --------
MylambdaFunction123 python3.8        128     600      659
 2020-01-10T03:20:56.390+0000 lambda
MylambdaFunction123 python3.8        128       5     1426
 2019-12-25T09:19:02.238+0000 lambda
MylambdaFunction123 python3.8        128       5     1426
 2019-12-25T09:39:36.779+0000 lambda
```

```
MylambdaFunction123 python3.8        128     600     1426
 2019-12-25T09:52:59.872+0000 lambda
```

- Pour API plus de détails, consultez la section ListVersionsByFunctionRéférence des AWS Tools for PowerShell applets de commande.

### `New-LMAlias`

L'exemple de code suivant montre comment utiliser`New-LMAlias`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un nouvel alias Lambda pour une version et une configuration de routage spécifiées afin de spécifier le pourcentage de demandes d'appel qu'il reçoit.

```
New-LMAlias -FunctionName "MylambdaFunction123" -
RoutingConfig_AdditionalVersionWeight @{Name="1";Value="0.6} -Description "Alias for
 version 4" -FunctionVersion 4 -Name "PowershellAlias"
```

- Pour API plus de détails, consultez la section CreateAliasRéférence des AWS Tools for PowerShell applets de commande.

### `Publish-LMFunction`

L'exemple de code suivant montre comment utiliser`Publish-LMFunction`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une nouvelle fonction C# (dotnetcore1.0 runtime) nommée dans MyFunction AWS Lambda, fournissant les fichiers binaires compilés pour la fonction à partir d'un fichier zip sur le système de fichiers local (des chemins relatifs ou absolus peuvent être utilisés). Les fonctions Lambda C# spécifient le gestionnaire de la fonction en utilisant la désignation : :Namespace. AssemblyName ClassName: :MethodName. Vous devez remplacer les parties du nom de l'assembly (sans le suffixe .dll), de l'espace de noms, du nom de classe et du nom de méthode de la spécification du gestionnaire de manière appropriée. La nouvelle fonction aura les variables d'environnement 'envvar1' et 'envvar2' configurées à partir des valeurs fournies.

```
Publish-LMFunction -Description "My C# Lambda Function" `
        -FunctionName MyFunction `
        -ZipFilename .\MyFunctionBinaries.zip `
```

```
        -Handler "AssemblyName::Namespace.ClassName::MethodName" `
        -Role "arn:aws:iam::123456789012:role/LambdaFullExecRole" `
        -Runtime dotnetcore1.0 `
        -Environment_Variable @{ "envvar1"="value";"envvar2"="value" }
```

Sortie :

```
CodeSha256       : /NgBMd...gq71I=
CodeSize         : 214784
DeadLetterConfig :
Description      : My C# Lambda Function
Environment      : Amazon.Lambda.Model.EnvironmentResponse
FunctionArn      : arn:aws:lambda:us-west-2:123456789012:function:ToUpper
FunctionName     : MyFunction
Handler          : AssemblyName::Namespace.ClassName::MethodName
KMSKeyArn        :
LastModified     : 2016-12-29T23:50:14.207+0000
MemorySize       : 128
Role             : arn:aws:iam::123456789012:role/LambdaFullExecRole
Runtime          : dotnetcore1.0
Timeout          : 3
Version          : $LATEST
VpcConfig        :
```

Exemple 2 : Cet exemple est similaire au précédent, sauf que les fichiers binaires des fonctions sont d'abord chargés dans un compartiment Amazon S3 (qui doit se trouver dans la même région que la fonction Lambda prévue) et que l'objet S3 obtenu est ensuite référencé lors de la création de la fonction.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key MyFunctionBinaries.zip -File .
\MyFunctionBinaries.zip
Publish-LMFunction -Description "My C# Lambda Function" `
        -FunctionName MyFunction `
        -BucketName amzn-s3-demo-bucket `
        -Key MyFunctionBinaries.zip `
        -Handler "AssemblyName::Namespace.ClassName::MethodName" `
        -Role "arn:aws:iam::123456789012:role/LambdaFullExecRole" `
        -Runtime dotnetcore1.0 `
        -Environment_Variable @{ "envvar1"="value";"envvar2"="value" }
```

- Pour API plus de détails, consultez la section CreateFunctionRéférence des AWS Tools for PowerShell applets de commande.

### `Publish-LMVersion`

L'exemple de code suivant montre comment utiliser`Publish-LMVersion`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une version pour l'instantané existant du code de fonction Lambda

```
Publish-LMVersion -FunctionName "MylambdaFunction123" -Description "Publishing
 Existing Snapshot of function code as a  new version through Powershell"
```

- Pour API plus de détails, consultez la section PublishVersionRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-LMAlias`

L'exemple de code suivant montre comment utiliser`Remove-LMAlias`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la fonction Lambda Alias mentionnée dans la commande.

```
Remove-LMAlias -FunctionName "MylambdaFunction123" -Name "NewAlias"
```

- Pour API plus de détails, consultez la section DeleteAliasRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-LMFunction`

L'exemple de code suivant montre comment utiliser`Remove-LMFunction`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime une version spécifique d'une fonction Lambda

```
Remove-LMFunction -FunctionName "MylambdaFunction123" -Qualifier '3'
```

- Pour API plus de détails, consultez la section DeleteFunctionRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-LMFunctionConcurrency`

L'exemple de code suivant montre comment utiliser`Remove-LMFunctionConcurrency`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la simultanéité des fonctions de la fonction Lambda.

```
Remove-LMFunctionConcurrency -FunctionName "MylambdaFunction123"
```

- Pour API plus de détails, consultez la section DeleteFunctionConcurrencyRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-LMPermission`

L'exemple de code suivant montre comment utiliser`Remove-LMPermission`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la politique de fonction pour la fonction Lambda spécifiée StatementId .

```
$policy =  Get-LMPolicy -FunctionName "MylambdaFunction123" -Select Policy |
 ConvertFrom-Json| Select-Object -ExpandProperty Statement
Remove-LMPermission -FunctionName "MylambdaFunction123" -StatementId $policy[0].Sid
```

- Pour API plus de détails, consultez la section RemovePermissionRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-LMProvisionedConcurrencyConfig`

L'exemple de code suivant montre comment utiliser`Remove-LMProvisionedConcurrencyConfig`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la configuration de simultanéité provisionnée pour un alias spécifique.

```
Remove-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -Qualifier
 "NewAlias1"
```

- Pour API plus de détails, consultez la section DeleteProvisionedConcurrencyConfigRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-LMResourceTag`

L'exemple de code suivant montre comment utiliser`Remove-LMResourceTag`.

Outils pour PowerShell

Exemple 1 : Supprime les balises fournies d'une fonction. L'applet de commande demandera une confirmation avant de continuer, sauf si le commutateur -Force est spécifié. Un seul appel est envoyé au service pour retirer les tags.

```
Remove-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction" -TagKey "Washington","Oregon","California"
```

Exemple 2 : Supprime les balises fournies d'une fonction. L'applet de commande demandera une confirmation avant de continuer, sauf si le commutateur -Force est spécifié. Une fois que l'appel au service est effectué, selon le tag fourni.

```
"Washington","Oregon","California" | Remove-LMResourceTag -Resource
 "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
```

- Pour API plus de détails, consultez la section UntagResourceRéférence des AWS Tools for PowerShell applets de commande.

### `Update-LMAlias`

L'exemple de code suivant montre comment utiliser`Update-LMAlias`.

Outils pour PowerShell

Exemple 1 : Cet exemple met à jour la configuration d'un alias de fonction Lambda existant. Il met à jour la RoutingConfiguration valeur pour transférer 60 % (0,6) du trafic vers la version 1

```
Update-LMAlias -FunctionName "MylambdaFunction123" -Description " Alias for version
 2" -FunctionVersion 2 -Name "newlabel1" -RoutingConfig_AdditionalVersionWeight
 @{Name="1";Value="0.6}
```

- Pour API plus de détails, consultez la section UpdateAliasRéférence des AWS Tools for PowerShell applets de commande.

### `Update-LMFunctionCode`

L'exemple de code suivant montre comment utiliser`Update-LMFunctionCode`.

Outils pour PowerShell

Exemple 1 : met à jour la fonction nommée MyFunction « » avec le nouveau contenu contenu dans le fichier zip spécifié. Pour un C#. NETFonction Lambda principale : le fichier zip doit contenir l'assemblage compilé.

```
Update-LMFunctionCode -FunctionName MyFunction -ZipFilename .\UpdatedCode.zip
```

Exemple 2 : Cet exemple est similaire au précédent mais utilise un objet Amazon S3 contenant le code mis à jour pour mettre à jour la fonction.

```
Update-LMFunctionCode -FunctionName MyFunction -BucketName amzn-s3-demo-bucket -Key
 UpdatedCode.zip
```

- Pour API plus de détails, consultez la section UpdateFunctionCodeRéférence des AWS Tools for PowerShell applets de commande.

### `Update-LMFunctionConfiguration`

L'exemple de code suivant montre comment utiliser`Update-LMFunctionConfiguration`.

Outils pour PowerShell

Exemple 1 : Cet exemple met à jour la configuration de la fonction Lambda existante

```
Update-LMFunctionConfiguration -FunctionName "MylambdaFunction123" -Handler
 "lambda_function.launch_instance" -Timeout 600 -Environment_Variable
 @{ "envvar1"="value";"envvar2"="value" } -Role arn:aws:iam::123456789101:role/
service-role/lambda -DeadLetterConfig_TargetArn arn:aws:sns:us-east-1:
 123456789101:MyfirstTopic
```

- Pour API plus de détails, consultez la section UpdateFunctionConfigurationRéférence des AWS Tools for PowerShell applets de commande.

### `Write-LMFunctionConcurrency`

L'exemple de code suivant montre comment utiliser`Write-LMFunctionConcurrency`.

Outils pour PowerShell

Exemple 1 : Cet exemple applique les paramètres de simultanéité pour la fonction dans son ensemble.

```
Write-LMFunctionConcurrency -FunctionName "MylambdaFunction123" -
ReservedConcurrentExecution 100
```

- Pour API plus de détails, consultez la section PutFunctionConcurrencyRéférence des AWS Tools for PowerShell applets de commande.

### `Write-LMProvisionedConcurrencyConfig`

L'exemple de code suivant montre comment utiliser`Write-LMProvisionedConcurrencyConfig`.

Outils pour PowerShell

Exemple 1 : cet exemple ajoute une configuration de simultanéité provisionnée à l'alias d'une fonction

```
Write-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -
ProvisionedConcurrentExecution 20 -Qualifier "NewAlias1"
```

- Pour API plus de détails, consultez la section PutProvisionedConcurrencyConfigRéférence des AWS Tools for PowerShell applets de commande.

## Exemples d'Amazon ML utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS Tools for PowerShell aide d'Amazon ML.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `Get-MLBatchPrediction`

L'exemple de code suivant montre comment utiliser`Get-MLBatchPrediction`.

Outils pour PowerShell

Exemple 1 : renvoie les métadonnées détaillées d'une prédiction par lots avec ID ID.

```
Get-MLBatchPrediction -BatchPredictionId ID
```

- Pour API plus de détails, consultez la section GetBatchPredictionRéférence des AWS Tools for PowerShell applets de commande.

### `Get-MLBatchPredictionList`

L'exemple de code suivant montre comment utiliser`Get-MLBatchPredictionList`.

Outils pour PowerShell

Exemple 1 : renvoie une liste de tous BatchPredictions les enregistrements de données associés qui correspondent au critère de recherche indiqué dans la demande.

```
Get-MLBatchPredictionList
```

Exemple 2 : renvoie une liste de toutes les BatchPredictions données dont le statut estCOMPLETED.

```
Get-MLBatchPredictionList -FilterVariable Status -EQ COMPLETED
```

- Pour API plus de détails, consultez la section DescribeBatchPredictionsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-MLDataSource`

L'exemple de code suivant montre comment utiliser`Get-MLDataSource`.

Outils pour PowerShell

Exemple 1 : renvoie les métadonnées, le statut et les informations du fichier de données pour un DataSource avec l'ID ID

```
Get-MLDataSource -DataSourceId ID
```

- Pour API plus de détails, consultez la section GetDataSourceRéférence des AWS Tools for PowerShell applets de commande.

### `Get-MLDataSourceList`

L'exemple de code suivant montre comment utiliser`Get-MLDataSourceList`.

Outils pour PowerShell

Exemple 1 : renvoie une liste de tous DataSources les enregistrements de données associés.

```
Get-MLDataSourceList
```

Exemple 2 : renvoie une liste de toutes les DataSources données dont le statut estCOMPLETED.

```
Get-MLDataDourceList -FilterVariable Status -EQ COMPLETED
```

- Pour API plus de détails, consultez la section DescribeDataSourcesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-MLEvaluation`

L'exemple de code suivant montre comment utiliser`Get-MLEvaluation`.

Outils pour PowerShell

Exemple 1 : renvoie les métadonnées et le statut d'une évaluation avec identifiant.

```
Get-MLEvaluation -EvaluationId ID
```

- Pour API plus de détails, consultez la section GetEvaluationRéférence des AWS Tools for PowerShell applets de commande.

### `Get-MLEvaluationList`

L'exemple de code suivant montre comment utiliser`Get-MLEvaluationList`.

Outils pour PowerShell

Exemple 1 : renvoie une liste de toutes les ressources d'évaluation

```
Get-MLEvaluationList
```

Exemple 2 : renvoie une liste de toutes les évaluations dont le statut est. COMPLETED

```
Get-MLEvaluationList -FilterVariable Status -EQ COMPLETED
```

- Pour API plus de détails, consultez la section DescribeEvaluationsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-MLModel`

L'exemple de code suivant montre comment utiliser`Get-MLModel`.

Outils pour PowerShell

Exemple 1 : renvoie les métadonnées détaillées, le statut, le schéma et les informations du fichier de données pour un identifiant MLModel avec identifiant.

```
Get-MLModel -ModelId ID
```

- Pour API plus de détails, voir G etMLModel dans le manuel de référence des AWS Tools for PowerShell applets de commande.

### `Get-MLModelList`

L'exemple de code suivant montre comment utiliser`Get-MLModelList`.

Outils pour PowerShell

Exemple 1 : renvoie une liste de tous les modèles et de leurs enregistrements de données associés.

```
Get-MLModelList
```

Exemple 2 : renvoie une liste de tous les modèles dont le statut estCOMPLETED.

```
Get-MLModelList -FilterVariable Status -EQ COMPLETED
```

- Pour API plus de détails, reportez-vous à la section D escribeMLModels dans la référence des AWS Tools for PowerShell applets de commande.

### `Get-MLPrediction`

L'exemple de code suivant montre comment utiliser`Get-MLPrediction`.

Outils pour PowerShell

Exemple 1 : envoyer un enregistrement au point de terminaison de prédiction en temps réel URL pour le modèle avec ID ID.

```
Get-MLPrediction -ModelId ID -PredictEndpoint URL -Record @{"A" = "B"; "C" = "D";}
```

- Pour API plus de détails, consultez la section Predict in AWS Tools for PowerShell Cmdlet Reference.

### `New-MLBatchPrediction`

L'exemple de code suivant montre comment utiliser`New-MLBatchPrediction`.

Outils pour PowerShell

Exemple 1 : créer une nouvelle demande de prédiction par lots pour le modèle avec ID ID et placer la sortie à l'emplacement S3 spécifié.

```
New-MLBatchPrediction -ModelId ID -Name NAME -OutputURI s3://...
```

- Pour API plus de détails, consultez la section CreateBatchPredictionRéférence des AWS Tools for PowerShell applets de commande.

### `New-MLDataSourceFromS3`

L'exemple de code suivant montre comment utiliser`New-MLDataSourceFromS3`.

Outils pour PowerShell

Exemple 1 : créer une source de données contenant les données d'un emplacement S3, avec un nom NAME et un schéma deSCHEMA.

```
New-MLDataSourceFromS3 -Name NAME -ComputeStatistics $true -DataSpec_DataLocationS3
 "s3://BUCKET/KEY" -DataSchema SCHEMA
```

- Pour API plus de détails, consultez CreateDataSourceFromS3 dans le manuel de référence des AWS Tools for PowerShell applets de commande.

### `New-MLEvaluation`

L'exemple de code suivant montre comment utiliser`New-MLEvaluation`.

Outils pour PowerShell

Exemple 1 : créer une évaluation pour un identifiant de source de données et un identifiant de modèle donnés

```
New-MLEvaluation -Name NAME -DataSourceId DSID -ModelId MID
```

- Pour API plus de détails, consultez la section CreateEvaluationRéférence des AWS Tools for PowerShell applets de commande.

### `New-MLModel`

L'exemple de code suivant montre comment utiliser`New-MLModel`.

Outils pour PowerShell

Exemple 1 : créer un nouveau modèle avec des données d'entraînement.

```
New-MLModel -Name NAME -ModelType BINARY -Parameter @{...} -TrainingDataSourceId ID
```

- Pour API plus de détails, reportez-vous à la section C reateMLModel dans le manuel de référence des AWS Tools for PowerShell applets de commande.

#### `New-MLRealtimeEndpoint`

L'exemple de code suivant montre comment utiliser`New-MLRealtimeEndpoint`.

Outils pour PowerShell

Exemple 1 : créer un nouveau point de terminaison de prédiction en temps réel pour l'identifiant de modèle donné.

```
New-MLRealtimeEndpoint -ModelId ID
```

- Pour API plus de détails, consultez la section CreateRealtimeEndpointRéférence des AWS Tools for PowerShell applets de commande.

## Exemples de Macie utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide de AWS Tools for PowerShell with Macie.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

### Actions

#### `Get-MAC2FindingList`

L'exemple de code suivant montre comment utiliser`Get-MAC2FindingList`.

Exemple 1 : Renvoie la liste FindingIds des résultats contenant une détection de données sensibles de type « CREDIT _ CARD _ NUMBER » ou « US_ _ SOCIAL SECURITY _ NUMBER »

```
$criterionAddProperties = New-Object
 Amazon.Macie2.Model.CriterionAdditionalProperties

$criterionAddProperties.Eq = @(
"CREDIT_CARD_NUMBER"
"US_SOCIAL_SECURITY_NUMBER"
)

$FindingCriterion = @{
'classificationDetails.result.sensitiveData.detections.type' =
 [Amazon.Macie2.Model.CriterionAdditionalProperties]$criterionAddProperties
}

Get-MAC2FindingList -FindingCriteria_Criterion $FindingCriterion -MaxResult 5
```

- Pour API plus de détails, consultez la section ListFindingsRéférence des AWS Tools for PowerShell applets de commande.

# AWS OpsWorks exemples utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS Tools for PowerShell with AWS OpsWorks.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `New-OPSDeployment`

L'exemple de code suivant montre comment utiliser`New-OPSDeployment`.

Outils pour PowerShell

Exemple 1 : Cette commande crée un nouveau déploiement d'applications sur toutes les instances basées sur Linux d'une couche dans Stacks. AWS OpsWorks Même si vous spécifiez un ID de couche, vous devez également spécifier un ID de pile. La commande permet au déploiement de redémarrer les instances si nécessaire.

```
New-OPSDeployment -StackID "724z93zz-zz78-4zzz-8z9z-1290123zzz1z" -LayerId
 "511b99c5-ec78-4caa-8a9d-1440116ffd1b" -AppId "0f7a109c-bf68-4336-8cb9-
d37fe0b8c61d" -Command_Name deploy -Command_Arg @{Name="allow_reboot";Value="true"}
```

Exemple 2 : Cette commande déploie la **`appsetup`** recette du livre de **`phpapp`** recettes et la **`secbaseline`** recette du livre de **`testcookbook`** recettes. La cible de déploiement est une instance, mais l'ID de pile et l'ID de couche sont également requis. L'**`allow_reboot`**attribut du paramètre Command_Arg est défini sur**`true`**, ce qui permet au déploiement de redémarrer les instances si nécessaire.

```
$commandArgs = '{ "Name":"execute_recipes", "Args"{ "recipes":
["phpapp::appsetup","testcookbook::secbaseline"] } }'
New-OPSDeployment -StackID "724z93zz-zz78-4zzz-8z9z-1290123zzz1z"
 -LayerId "511b99c5-ec78-4caa-8a9d-1440116ffd1b" -InstanceId
 "d89a6118-0007-4ccf-a51e-59f844127021" -Command_Name $commandArgs -Command_Arg
 @{Name="allow_reboot";Value="true
```

- Pour API plus de détails, consultez la section CreateDeploymentRéférence des AWS Tools for PowerShell applets de commande.

# AWS Price List exemples utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS Tools for PowerShell with AWS Price List.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `Get-PLSAttributeValue`

L'exemple de code suivant montre comment utiliser`Get-PLSAttributeValue`.

Outils pour PowerShell

Exemple 1 : renvoie les valeurs de l'attribut « volumeType » pour Amazon EC2 dans la région us-east-1.

```
Get-PLSAttributeValue -ServiceCode AmazonEC2 -AttributeName "volumeType" -region us-
east-1
```

Sortie :

```
Value
-----
Cold HDD
General Purpose
Magnetic
Provisioned IOPS
Throughput Optimized HDD
```

- Pour API plus de détails, consultez la section GetAttributeValuesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-PLSProduct`

L'exemple de code suivant montre comment utiliser`Get-PLSProduct`.

Outils pour PowerShell

Exemple 1 : Retourne les détails de tous les produits pour AmazonEC2.

```
Get-PLSProduct -ServiceCode AmazonEC2 -Region us-east-1
```

Sortie :

```
{"product":{"productFamily":"Compute Instance","attributes":
{"enhancedNetworkingSupported":"Yes","memory":"30.5
 GiB","dedicatedEbsThroughput":"800 Mbps","vcpu":"4","locationType":"AWS
 Region","storage":"EBS only","instanceFamily":"Memory
 optimized","operatingSystem":"SUSE","physicalProcessor":"Intel Xeon E5-2686 v4
 (Broadwell)","clockSpeed":"2.3 GHz","ecu":"Variable","networkPerformance":"Up
 to 10 Gigabit","servicename":"Amazon Elastic Compute
 Cloud","instanceType":"r4.xlarge","tenancy":"Shared","usagetype":"USW2-
BoxUsage:r4.xlarge","normalizationSizeFactor":"8","processorFeatures":"Intel AVX,
 Intel AVX2, Intel Turbo","servicecode":"AmazonEC2","licenseModel":"No License
 required","currentGeneration":"Yes","preInstalledSw":"NA","location":"US West
 (Oregon)","processorArchitecture":"64-bit","operation":"RunInstances:000g"},...
```

Exemple 2 : les données de retour pour Amazon EC2 dans la région us-east-1 sont filtrées par types de volumes « à usage général » garantis par des types de volumes. SSD

```
Get-PLSProduct -ServiceCode AmazonEC2 -Filter
 @{Type="TERM_MATCH";Field="volumeType";Value="General
 Purpose"},@{Type="TERM_MATCH";Field="storageMedia";Value="SSD-backed"} -Region us-
east-1
```

Sortie :

```
{"product":{"productFamily":"Storage","attributes":{"storageMedia":"SSD-
backed","maxThroughputvolume":"160 MB/sec","volumeType":"General
 Purpose","maxIopsvolume":"10000",...
```

- Pour API plus de détails, consultez la section GetProductsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-PLSService`

L'exemple de code suivant montre comment utiliser`Get-PLSService`.

Exemple 1 : renvoie les métadonnées de tous les codes de service disponibles dans la région us-east-1.

```
Get-PLSService -Region us-east-1
```

Sortie :

```
AttributeNames                                                  ServiceCode
--------------                                                  -----------
{productFamily, servicecode, groupDescription, termType...}     AWSBudgets
{productFamily, servicecode, termType, usagetype...}            AWSCloudTrail
{productFamily, servicecode, termType, usagetype...}            AWSCodeCommit
{productFamily, servicecode, termType, usagetype...}            AWSCodeDeploy
{productFamily, servicecode, termType, usagetype...}            AWSCodePipeline
{productFamily, servicecode, termType, usagetype...}            AWSConfig
...
```

Exemple 2 : renvoie les métadonnées du EC2 service Amazon dans la région us-east-1.

```
Get-PLSService -ServiceCode AmazonEC2 -Region us-east-1
```

Sortie :

```
AttributeNames                                                          ServiceCode
--------------                                                          -----------
{volumeType, maxIopsvolume, instanceCapacity10xlarge, locationType...} AmazonEC2
```

- Pour API plus de détails, consultez la section DescribeServicesRéférence des AWS Tools for PowerShell applets de commande.

# Exemples de groupes de ressources utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide de AWS Tools for PowerShell with Resource Groups.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `Add-RGResourceTag`

L'exemple de code suivant montre comment utiliser`Add-RGResourceTag`.

Outils pour PowerShell

Exemple 1 : Cet exemple ajoute la clé de balise « Instances » avec la valeur « workboxes » au groupe de ressources donné arn

```
Add-RGResourceTag -Tag @{Instances="workboxes"} -Arn arn:aws:resource-groups:eu-
west-1:123456789012:group/workboxes
```

Sortie :

```
Arn                                                              Tags
---                                                              ----
arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes {[Instances,
 workboxes]}
```

- Pour API plus de détails, consultez la section Référence des balises dans les AWS Tools for PowerShell applets de commande.

### `Find-RGResource`

L'exemple de code suivant montre comment utiliser`Find-RGResource`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un type de ressource ResourceQuery for Instance avec des filtres de balises et trouve des ressources.

```
$query = [Amazon.ResourceGroups.Model.ResourceQuery]::new()
$query.Type = [Amazon.ResourceGroups.QueryType]::TAG_FILTERS_1_0
$query.Query = ConvertTo-Json -Compress -Depth 4 -InputObject @{
  ResourceTypeFilters = @('AWS::EC2::Instance')
  TagFilters = @(@{
    Key = 'auto'
    Values = @('no')
  })
 }

Find-RGResource -ResourceQuery $query | Select-Object -ExpandProperty
 ResourceIdentifiers
```

Sortie :

```
ResourceArn                                                  ResourceType
-----------                                                  ------------
arn:aws:ec2:eu-west-1:123456789012:instance/i-0123445b6cb7bd67b AWS::EC2::Instance
```

- Pour API plus de détails, consultez la section SearchResourcesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-RGGroup`

L'exemple de code suivant montre comment utiliser`Get-RGGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple récupère le groupe de ressources selon le nom du groupe

```
Get-RGGroup -GroupName auto-no
```

Sortie :

```
Description GroupArn                                                Name
----------- --------                                                ----
```

```
            arn:aws:resource-groups:eu-west-1:123456789012:group/auto-no auto-no
```

- Pour API plus de détails, consultez la section GetGroupRéférence des AWS Tools for PowerShell applets de commande.

### `Get-RGGroupList`

L'exemple de code suivant montre comment utiliser`Get-RGGroupList`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie les groupes de ressources déjà créés.

```
Get-RGGroupList
```

Sortie :

```
GroupArn                                                                GroupName
--------                                                                ---------
arn:aws:resource-groups:eu-west-1:123456789012:group/auto-no            auto-no
arn:aws:resource-groups:eu-west-1:123456789012:group/auto-yes           auto-yes
arn:aws:resource-groups:eu-west-1:123456789012:group/build600           build600
```

- Pour API plus de détails, consultez la section ListGroupsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-RGGroupQuery`

L'exemple de code suivant montre comment utiliser`Get-RGGroupQuery`.

Outils pour PowerShell

Exemple 1 : Cet exemple extrait la requête de ressource pour le groupe de ressources donné

```
Get-RGGroupQuery -GroupName auto-no | Select-Object -ExpandProperty ResourceQuery
```

Sortie :

```
Query
         Type
```

```
-----
         ----
{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":[{"Key":"auto","Values":
["no"]}]} TAG_FILTERS_1_0
```

- Pour API plus de détails, consultez la section GetGroupQueryRéférence des AWS Tools for PowerShell applets de commande.

### `Get-RGGroupResourceList`

L'exemple de code suivant montre comment utiliser`Get-RGGroupResourceList`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie les ressources du groupe sur la base du filtrage par type de ressource

```
Get-RGGroupResourceList -Filter @{Name="resource-type";Values="AWS::EC2::Instance"}
 -GroupName auto-yes | Select-Object -ExpandProperty ResourceIdentifiers
```

Sortie :

```
ResourceArn                                                    ResourceType
-----------                                                    ------------
arn:aws:ec2:eu-west-1:123456789012:instance/i-0123bc45b567890e1 AWS::EC2::Instance
arn:aws:ec2:eu-west-1:123456789012:instance/i-0a1caf2345f67d8dc AWS::EC2::Instance
arn:aws:ec2:eu-west-1:123456789012:instance/i-012e3cb4df567e8aa AWS::EC2::Instance
arn:aws:ec2:eu-west-1:123456789012:instance/i-0fd12dd3456789012 AWS::EC2::Instance
```

- Pour API plus de détails, consultez la section ListGroupResourcesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-RGResourceTag`

L'exemple de code suivant montre comment utiliser`Get-RGResourceTag`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie les balises pour le groupe de ressources arn donné

```
Get-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/
workboxes
```

Sortie :

```
Key       Value
---       -----
Instances workboxes
```

- Pour API plus de détails, consultez la section GetTagsRéférence des AWS Tools for PowerShell applets de commande.

### `New-RGGroup`

L'exemple de code suivant montre comment utiliser`New-RGGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un nouveau groupe de AWS ressources Resource Groups basé sur des balises nommé TestPowerShellGroup. Le groupe inclut les EC2 instances Amazon de la région actuelle qui sont étiquetées avec la clé de balise « Name » et la valeur de balise « test2 ». La commande renvoie la requête et le type de groupe, ainsi que les résultats de l'opération.

```
$ResourceQuery = New-Object -TypeName Amazon.ResourceGroups.Model.ResourceQuery
$ResourceQuery.Type = "TAG_FILTERS_1_0"
$ResourceQuery.Query = '{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":
[{"Key":"Name","Values":["test2"]}]}'
$ResourceQuery

New-RGGroup -Name TestPowerShellGroup -ResourceQuery $ResourceQuery -Description
 "Test resource group."
```

Sortie :

```
Query
            Type
-----
            ----
{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":[{"Key":"Name","Values":
["test2"]}]} TAG_FILTERS_1_0
```

```
LoggedAt         : 11/20/2018 2:40:59 PM
Group            : Amazon.ResourceGroups.Model.Group
ResourceQuery    : Amazon.ResourceGroups.Model.ResourceQuery
Tags             : {}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 338
HttpStatusCode   : OK
```

- Pour API plus de détails, consultez la section CreateGroupRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-RGGroup`

L'exemple de code suivant montre comment utiliser`Remove-RGGroup`.

Outils pour PowerShell

Exemple 1 : cet exemple supprime le groupe de ressources nommé

```
Remove-RGGroup -GroupName non-tag-cfn-elbv2
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-RGGroup (DeleteGroup)" on target "non-tag-cfn-
elbv2".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

Description GroupArn
 Name
----------- --------
 ----
           arn:aws:resource-groups:eu-west-1:123456789012:group/non-tag-cfn-elbv2
 non-tag-cfn-elbv2
```

- Pour API plus de détails, consultez la section DeleteGroupRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-RGResourceTag`

L'exemple de code suivant montre comment utiliser`Remove-RGResourceTag`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la balise mentionnée du groupe de ressources

```
Remove-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/
workboxes -Key Instances
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-RGResourceTag (Untag)" on target "arn:aws:resource-
groups:eu-west-1:933303704102:group/workboxes".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

Arn                                                          Keys
---                                                          ----
arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes {Instances}
```

- Pour API plus de détails, voir Untag in AWS Tools for PowerShell Cmdlet Reference.

### `Update-RGGroup`

L'exemple de code suivant montre comment utiliser`Update-RGGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple met à jour la description du groupe

```
Update-RGGroup -GroupName auto-yes -Description "Instances auto-remove"
```

Sortie :

```
Description             GroupArn
  Name
-----------             --------
  ----
```

```
Instances to be cleaned arn:aws:resource-groups:eu-west-1:123456789012:group/auto-
yes auto-yes
```

- Pour API plus de détails, consultez la section UpdateGroupRéférence des AWS Tools for PowerShell applets de commande.

### `Update-RGGroupQuery`

L'exemple de code suivant montre comment utiliser`Update-RGGroupQuery`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un objet de requête et met à jour la requête pour le groupe.

```
$query = [Amazon.ResourceGroups.Model.ResourceQuery]::new()
$query.Type = [Amazon.ResourceGroups.QueryType]::TAG_FILTERS_1_0
$query.Query = @{
  ResourceTypeFilters = @('AWS::EC2::Instance')
  TagFilters = @(@{
  Key='Environment'
  Values='Build600.11'
  })
} | ConvertTo-Json -Compress -Depth 4

Update-RGGroupQuery -GroupName build600 -ResourceQuery $query
```

Sortie :

```
GroupName ResourceQuery
--------- -------------
build600  Amazon.ResourceGroups.Model.ResourceQuery
```

- Pour API plus de détails, consultez la section UpdateGroupQueryRéférence des AWS Tools for PowerShell applets de commande.

# Resource Groups : API exemples de balisage à l'aide de Tools for PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du balisage AWS Tools for PowerShell API with Resource Groups.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques



## Actions

### `Add-RGTResourceTag`

L'exemple de code suivant montre comment utiliser`Add-RGTResourceTag`.

Outils pour PowerShell

Exemple 1 : cet exemple ajoute les clés de balise « stage » et « version » avec les valeurs « beta » et « preprod_test » à un compartiment Amazon S3 et à une table Amazon DynamoDB. Un seul appel est envoyé au service pour appliquer les balises.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

Add-RGTResourceTag -ResourceARNList $arn1,$arn2 -Tag @{ "stage"="beta";
 "version"="preprod_test" }
```

Exemple 2 : Cet exemple ajoute les balises et les valeurs spécifiées à un compartiment Amazon S3 et à une table Amazon DynamoDB. Deux appels sont effectués vers le service, un pour chaque ressource redirigée vers ARN l'applet de commande.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

$arn1,$arn2 | Add-RGTResourceTag -Tag @{ "stage"="beta"; "version"="preprod_test" }
```

- Pour API plus de détails, consultez la section TagResourcesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-RGTResource`

L'exemple de code suivant montre comment utiliser`Get-RGTResource`.

Outils pour PowerShell

Exemple 1 : renvoie toutes les ressources balisées d'une région et les clés de balise associées à la ressource. Si aucun paramètre -Region n'est fourni à l'applet de commande, elle tentera de déduire la région à partir des métadonnées du shell ou de l'instance. EC2

```
Get-RGTResource
```

Sortie :

```
ResourceARN                                                          Tags
-----------                                                          ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable                {stage, version}
arn:aws:s3:::mybucket                                                {stage, version,
 othertag}
```

Exemple 2 : renvoie toutes les ressources balisées du type spécifié dans une région. La chaîne pour chaque nom de service et type de ressource est identique à celle incorporée dans le nom de ressource Amazon (ARN) d'une ressource.

```
Get-RGTResource -ResourceType "s3"
```

Sortie :

```
ResourceARN                                                          Tags
-----------                                                          ----
arn:aws:s3:::mybucket                                                {stage, version,
 othertag}
```

Exemple 3 : renvoie toutes les ressources balisées du type spécifié dans une région. Notez que lorsque les types de ressources sont redirigés vers l'applet de commande, un appel au service est effectué pour chaque type de ressource fourni.

```
"dynamodb","s3" | Get-RGTResource
```

Sortie :

```
ResourceARN                                                          Tags
-----------                                                          ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable                {stage, version}
arn:aws:s3:::mybucket                                                {stage, version,
 othertag}
```

Exemple 4 : renvoie toutes les ressources balisées correspondant au filtre spécifié.

```
Get-RGTResource -TagFilter @{ Key="stage" }
```

Sortie :

```
ResourceARN                                                          Tags
-----------                                                          ----
arn:aws:s3:::mybucket                                                {stage, version,
 othertag}
```

Exemple 5 : renvoie toutes les ressources balisées correspondant au filtre et au type de ressource spécifiés.

```
Get-RGTResource -TagFilter @{ Key="stage" } -ResourceType "dynamodb"
```

Sortie :

```
ResourceARN                                                          Tags
-----------                                                          ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable                {stage, version}
```

Exemple 6 : renvoie toutes les ressources balisées correspondant au filtre spécifié.

```
Get-RGTResource -TagFilter @{ Key="stage"; Values=@("beta","gamma") }
```

Sortie :

```
ResourceARN                                                          Tags
-----------                                                          ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable                {stage, version}
```

- Pour API plus de détails, consultez la section GetResourcesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-RGTTagKey`

L'exemple de code suivant montre comment utiliser`Get-RGTTagKey`.

Outils pour PowerShell

Exemple 1 : renvoie toutes les clés de balise de la région spécifiée. Si le paramètre -Region n'est pas spécifié, l'applet de commande tentera de déduire la région à partir de la région shell par défaut ou des métadonnées de l'instance. EC2 Notez que les clés du tag ne sont pas renvoyées dans un ordre spécifique.

```
Get-RGTTagKey -region us-west-2
```

Sortie :

```
version
stage
```

- Pour API plus de détails, consultez la section GetTagKeysRéférence des AWS Tools for PowerShell applets de commande.

### `Get-RGTTagValue`

L'exemple de code suivant montre comment utiliser`Get-RGTTagValue`.

Outils pour PowerShell

Exemple 1 : renvoie la valeur de la balise spécifiée dans une région. Si le paramètre -Region n'est pas spécifié, l'applet de commande tentera de déduire la région à partir de la région shell par défaut ou des métadonnées de l'instance. EC2

```
Get-RGTTagValue -Key "stage" -Region us-west-2
```

Sortie :

```
beta
```

- Pour API plus de détails, consultez la section GetTagValuesRéférence des AWS Tools for PowerShell applets de commande.

**`Remove-RGTResourceTag`**

L'exemple de code suivant montre comment utiliser`Remove-RGTResourceTag`.

Outils pour PowerShell

Exemple 1 : Supprime les clés de balise « stage » et « version », ainsi que les valeurs associées, d'un compartiment Amazon S3 et d'une table Amazon DynamoDB. Un seul appel est envoyé au service pour retirer les tags. Avant de supprimer les balises, l'applet de commande vous invite à confirmer. Pour contourner la confirmation, ajoutez le paramètre -Force.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

Remove-RGTResourceTag -ResourceARNList $arn1,$arn2 -TagKey "stage","version"
```

Exemple 2 : Supprime les clés de balise « stage » et « version », ainsi que les valeurs associées, d'un compartiment Amazon S3 et d'une table Amazon DynamoDB. Deux appels sont effectués vers le service, un pour chaque ressource redirigée vers ARN l'applet de commande. Avant chaque appel, l'applet de commande demande une confirmation. Pour contourner la confirmation, ajoutez le paramètre -Force.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

$arn1,$arn2 | Remove-RGTResourceTag -TagKey "stage","version"
```

- Pour API plus de détails, consultez la section UntagResourcesRéférence des AWS Tools for PowerShell applets de commande.

# Exemples de Route 53 utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS Tools for PowerShell aide de Route 53.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `Edit-R53ResourceRecordSet`

L'exemple de code suivant montre comment utiliser`Edit-R53ResourceRecordSet`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un enregistrement A pour www.exemple.com et modifie l'enregistrement A pour test.exemple.com de 192.0.2.3 à 192.0.2.1. Notez que les valeurs des enregistrements TXT de type modifications doivent être entre guillemets. Consultez la documentation Amazon Route 53 pour plus de détails. Vous pouvez utiliser l'Get-R53Changeapplet de commande pour effectuer un sondage afin de déterminer quand les modifications sont terminées.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "www.example.com"
$change1.ResourceRecordSet.Type = "TXT"
$change1.ResourceRecordSet.TTL = 600
$change1.ResourceRecordSet.ResourceRecords.Add(@{Value="item 1 item 2 item 3"})

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "DELETE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "test.example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.TTL = 600
$change2.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.3"})

$change3 = New-Object Amazon.Route53.Model.Change
$change3.Action = "CREATE"
$change3.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change3.ResourceRecordSet.Name = "test.example.com"
```

```
$change3.ResourceRecordSet.Type = "A"
$change3.ResourceRecordSet.TTL = 600
$change3.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.1"})

$params = @{
    HostedZoneId="Z1PA6795UKMFR9"
 ChangeBatch_Comment="This change batch creates a TXT record for www.example.com.
 and changes the A record for test.example.com. from 192.0.2.3 to 192.0.2.1."
 ChangeBatch_Change=$change1,$change2,$change3
}

Edit-R53ResourceRecordSet @params
```

Exemple 2 : Cet exemple montre comment créer des ensembles d'enregistrements de ressources d'alias. « Z222222222 » est l'ID de la zone hébergée Amazon Route 53 dans laquelle vous créez le jeu d'enregistrements de ressources d'alias. « exemple.com » est l'apex de zone pour lequel vous souhaitez créer un alias et « www.exemple.com » est un sous-domaine pour lequel vous souhaitez également créer un alias. « Z1111111111111 » est un exemple d'ID de zone hébergée pour l'équilibreur de charge et « example-load-balancer -1111111111.us-east-1.elb.amazonaws.com » est un exemple de nom de domaine d'équilibreur de charge avec lequel Amazon Route 53 répond aux requêtes pour exemple.com et www.example.com. Consultez la documentation Amazon Route 53 pour plus de détails. Vous pouvez utiliser l'Get-R53Changeapplet de commande pour effectuer un sondage afin de déterminer quand les modifications sont terminées.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "www.example.com"
$change1.ResourceRecordSet.Type = "A"
```

```
$change1.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $false

$params = @{
    HostedZoneId="Z222222222"
 ChangeBatch_Comment="This change batch creates two alias resource record sets, one
 for the zone apex, example.com, and one for www.example.com, that both point to
 example-load-balancer-1111111111.us-east-1.elb.amazonaws.com."
 ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

Exemple 3 : Cet exemple crée deux enregistrements A pour www.example.com. Un quart du temps (1/ (1+3)), Amazon Route 53 répond aux requêtes relatives à www.example.com avec les deux valeurs du premier ensemble d'enregistrements de ressources (192.0.2.9 et 192.0.2.10). Dans les trois quarts des cas (3/ (1+3)), Amazon Route 53 répond aux requêtes relatives à www.example.com avec les deux valeurs du deuxième ensemble d'enregistrements de ressources (192.0.2.11 et 192.0.2.12). Consultez la documentation Amazon Route 53 pour plus de détails. Vous pouvez utiliser l'Get-R53Changeapplet de commande pour effectuer un sondage afin de déterminer quand les modifications sont terminées.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "www.example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "Rack 2, Positions 4 and 5"
$change1.ResourceRecordSet.Weight = 1
$change1.ResourceRecordSet.TTL = 600
$change1.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.9"})
$change1.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.10"})

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "www.example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "Rack 5, Positions 1 and 2"
```

```
$change2.ResourceRecordSet.Weight = 3
$change2.ResourceRecordSet.TTL = 600
$change2.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.11"})
$change2.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.12"})

$params = @{
    HostedZoneId="Z1PA6795UKMFR9"
    ChangeBatch_Comment="This change creates two weighted resource record sets, each
 of which has two values."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

Exemple 4 : Cet exemple montre comment créer des ensembles d'enregistrements de ressources d'alias pondérés en supposant que exemple.com est le domaine pour lequel vous souhaitez créer des ensembles d'enregistrements de ressources d'alias pondérés. SetIdentifier différencie les deux ensembles d'enregistrements de ressources alias pondérés l'un de l'autre. Cet élément est obligatoire car les éléments Nom et Type ont les mêmes valeurs pour les deux ensembles d'enregistrements de ressources. Z1111111111111 et Z3333333333333 sont des exemples de zone hébergée pour l'équilibreur de charge spécifié par la valeur de. IDs ELB DNSName example-load-balancer-2222222222.us-east-1.elb.amazonaws.com et example-load-balancer -4444444444.us-east-1.elb.amazonaws.com sont des exemples de domaines Elastic Load Balancing à partir desquels Amazon Route 53 répond aux requêtes pour exemple.com. Consultez la documentation Amazon Route 53 pour plus de détails. Vous pouvez utiliser l'Get-R53Changeapplet de commande pour effectuer un sondage afin de déterminer quand les modifications sont terminées.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "1"
$change1.ResourceRecordSet.Weight = 3
$change1.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-2222222222.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true
```

```
$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "2"
$change2.ResourceRecordSet.Weight = 1
$change2.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change2.ResourceRecordSet.AliasTarget.HostedZoneId = "Z3333333333333"
$change2.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-4444444444.us-east-1.elb.amazonaws.com."
$change2.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $false

$params = @{
    HostedZoneId="Z5555555555"
    ChangeBatch_Comment="This change batch creates two weighted alias resource
 record sets. Amazon Route 53 responds to queries for example.com with the first ELB
 domain 3/4ths of the times and the second one 1/4th of the time."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

Exemple 5 : Cet exemple crée deux ensembles d'enregistrements de ressources d'alias de latence, l'un pour un équilibreur de ELB charge dans la région USA Ouest (Oregon) (us-west-2) et l'autre pour un équilibreur de charge dans la région Asie-Pacifique (Singapour) (ap-southeast-1). Consultez la documentation Amazon Route 53 pour plus de détails. Vous pouvez utiliser l'Get-R53Changeapplet de commande pour effectuer un sondage afin de déterminer quand les modifications sont terminées.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "Oregon load balancer 1"
$change1.ResourceRecordSet.Region = us-west-2
$change1.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-2222222222.us-west-2.elb.amazonaws.com"
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true
```

```
$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "Singapore load balancer 1"
$change2.ResourceRecordSet.Region = ap-southeast-1
$change2.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change2.ResourceRecordSet.AliasTarget.HostedZoneId = "Z2222222222222"
$change2.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.ap-southeast-1.elb.amazonaws.com"
$change2.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$params = @{
    HostedZoneId="Z5555555555"
    ChangeBatch_Comment="This change batch creates two latency resource record
 sets, one for the US West (Oregon) region and one for the Asia Pacific (Singapore)
 region."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

- Pour API plus de détails, consultez la section ChangeResourceRecordSetsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-R53AccountLimit`

L'exemple de code suivant montre comment utiliser`Get-R53AccountLimit`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie le nombre maximum de zones hébergées pouvant être créées à l'aide du compte actuel.

```
Get-R53AccountLimit -Type MAX_HOSTED_ZONES_BY_OWNER
```

Sortie :

```
15
```

- Pour API plus de détails, consultez la section GetAccountLimitRéférence des AWS Tools for PowerShell applets de commande.

### `Get-R53CheckerIpRanges`

L'exemple de code suivant montre comment utiliser`Get-R53CheckerIpRanges`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie le CIDRs pour les vérificateurs de santé Route53

```
Get-R53CheckerIpRanges
```

Sortie :

```
15.177.2.0/23
15.177.6.0/23
15.177.10.0/23
15.177.14.0/23
15.177.18.0/23
15.177.22.0/23
15.177.26.0/23
15.177.30.0/23
15.177.34.0/23
15.177.38.0/23
15.177.42.0/23
15.177.46.0/23
15.177.50.0/23
15.177.54.0/23
15.177.58.0/23
15.177.62.0/23
54.183.255.128/26
54.228.16.0/26
54.232.40.64/26
54.241.32.64/26
54.243.31.192/26
54.244.52.192/26
54.245.168.0/26
54.248.220.0/26
54.250.253.192/26
54.251.31.128/26
54.252.79.128/26
```

```
54.252.254.192/26
54.255.254.192/26
107.23.255.0/26
176.34.159.192/26
177.71.207.128/26
```

- Pour API plus de détails, consultez la section GetCheckerIpRangesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-R53HostedZone`

L'exemple de code suivant montre comment utiliser`Get-R53HostedZone`.

Outils pour PowerShell

Exemple 1 : Renvoie les détails de la zone hébergée avec l'ID PJN98FT9 Z1D633.

```
Get-R53HostedZone -Id Z1D633PJN98FT9
```

- Pour API plus de détails, consultez la section GetHostedZoneRéférence des AWS Tools for PowerShell applets de commande.

### `Get-R53HostedZoneCount`

L'exemple de code suivant montre comment utiliser`Get-R53HostedZoneCount`.

Outils pour PowerShell

Exemple 1 : renvoie le nombre total de zones hébergées publiques et privées pour le moment Compte AWS.

```
Get-R53HostedZoneCount
```

- Pour API plus de détails, consultez la section GetHostedZoneCountRéférence des AWS Tools for PowerShell applets de commande.

### `Get-R53HostedZoneLimit`

L'exemple de code suivant montre comment utiliser`Get-R53HostedZoneLimit`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie la limite du nombre maximum d'enregistrements pouvant être créés dans la zone hébergée spécifiée.

```
Get-R53HostedZoneLimit -HostedZoneId Z3MEQ8T7HAAAAF -Type MAX_RRSETS_BY_ZONE
```

Sortie :

```
5
```

- Pour API plus de détails, consultez la section GetHostedZoneLimitRéférence des AWS Tools for PowerShell applets de commande.

### `Get-R53HostedZoneList`

L'exemple de code suivant montre comment utiliser`Get-R53HostedZoneList`.

Outils pour PowerShell

Exemple 1 : affiche toutes vos zones hébergées publiques et privées.

```
Get-R53HostedZoneList
```

Exemple 2 : affiche toutes les zones hébergées associées à l'ensemble de délégation réutilisable doté de l'ID NZ8X2CISAMPLE

```
Get-R53HostedZoneList -DelegationSetId NZ8X2CISAMPLE
```

- Pour API plus de détails, consultez la section ListHostedZonesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-R53HostedZonesByName`

L'exemple de code suivant montre comment utiliser`Get-R53HostedZonesByName`.

Outils pour PowerShell

Exemple 1 : renvoie toutes vos zones hébergées publiques et privées dans l'ASCIIordre par nom de domaine.

```
Get-R53HostedZonesByName
```

Exemple 2 : Renvoie vos zones hébergées publiques et privées, dans l'ASCIIordre par nom de domaine, en commençant par le DNS nom spécifié.

```
Get-R53HostedZonesByName -DnsName example2.com
```

Exemple 3 : Cet exemple montre comment énumérer manuellement les zones hébergées en récupérant d'abord un seul élément, puis en itérant deux à la fois jusqu'à ce que toutes les zones soient renvoyées, en utilisant les propriétés de marqueur associées à la réponse du service dans la **`$AWSHistory`** pile après chaque appel.

```
Get-R53HostedZonesByName -MaxItem 1
while ($LastServiceResponse.IsTruncated)
{
    $nextPageParams = @{
        DnsName=$LastServiceResponse.NextDNSName
        HostedZoneId=$LastServiceResponse.NextHostedZoneId
    }
    Get-R53HostedZonesByName -MaxItem 2 @nextPageParams
}
```

- Pour API plus de détails, consultez la section ListHostedZonesByNameRéférence des AWS Tools for PowerShell applets de commande.

### `Get-R53QueryLoggingConfigList`

L'exemple de code suivant montre comment utiliser`Get-R53QueryLoggingConfigList`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie toutes les configurations pour la journalisation des DNS requêtes associées au courant Compte AWS.

```
Get-R53QueryLoggingConfigList
```

Sortie :

```
Id                                   HostedZoneId   CloudWatchLogsLogGroupArn
--                                   ------------   -------------------------
```

```
59b0fa33-4fea-4471-a88c-926476aaa40d Z385PDS6EAAAZR arn:aws:logs:us-
east-1:111111111112:log-group:/aws/route53/example1.com:*
ee528e95-4e03-4fdc-9d28-9e24ddaaa063 Z94SJHBV1AAAAZ arn:aws:logs:us-
east-1:111111111112:log-group:/aws/route53/example2.com:*
e38dddda-ceb6-45c1-8cb7-f0ae56aaaa2b Z3MEQ8T7AAA1BF arn:aws:logs:us-
east-1:111111111112:log-group:/aws/route53/example3.com:*
```

- Pour API plus de détails, consultez la section ListQueryLoggingConfigsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-R53ReusableDelegationSet`

L'exemple de code suivant montre comment utiliser`Get-R53ReusableDelegationSet`.

Outils pour PowerShell

Exemple 1 : Cet exemple extrait des informations sur l'ensemble de délégations spécifié, y compris les quatre serveurs de noms assignés à l'ensemble de délégations.

```
Get-R53ReusableDelegationSet -Id N23DS9X4AYEAAA
```

Sortie :

```
Id                          CallerReference NameServers
--                          --------------- -----------
/delegationset/N23DS9X4AYEAAA testcaller      {ns-545.awsdns-04.net,
 ns-1264.awsdns-30.org, ns-2004.awsdns-58.co.uk, ns-240.awsdns-30.com}
```

- Pour API plus de détails, consultez la section GetReusableDelegationSetRéférence des AWS Tools for PowerShell applets de commande.

### `New-R53HostedZone`

L'exemple de code suivant montre comment utiliser`New-R53HostedZone`.

Outils pour PowerShell

Exemple 1 : crée une nouvelle zone hébergée nommée « exemple.com », associée à un ensemble de délégations réutilisable. Notez que vous devez fournir une valeur pour le CallerReference paramètre afin que les demandes puissent être réessayées si nécessaire

sans risquer d'exécuter l'opération deux fois. Comme la zone hébergée est créée dans un, VPC elle est automatiquement privée et vous ne devez pas définir le PrivateZone paramètre - HostedZoneConfig _.

```
$params = @{
    Name="example.com"
    CallerReference="myUniqueIdentifier"
    HostedZoneConfig_Comment="This is my first hosted zone"
    DelegationSetId="NZ8X2CISAMPLE"
    VPC_VPCId="vpc-1a2b3c4d"
    VPC_VPCRegion="us-east-1"
}

New-R53HostedZone @params
```

- Pour API plus de détails, consultez la section CreateHostedZoneRéférence des AWS Tools for PowerShell applets de commande.

### `New-R53QueryLoggingConfig`

L'exemple de code suivant montre comment utiliser`New-R53QueryLoggingConfig`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une nouvelle configuration de journalisation des DNS requêtes Route53 pour la zone hébergée spécifiée. Amazon Route53 publiera les journaux de DNS requêtes dans le groupe de journaux Cloudwatch spécifié.

```
New-R53QueryLoggingConfig -HostedZoneId Z3MEQ8T7HAAAAF -CloudWatchLogsLogGroupArn
 arn:aws:logs:us-east-1:111111111111:log-group:/aws/route53/example.com:*
```

Sortie :

```
QueryLoggingConfig                      Location
------------------                      --------
Amazon.Route53.Model.QueryLoggingConfig https://route53.amazonaws.com/2013-04-01/
queryloggingconfig/ee5aaa95-4e03-4fdc-9d28-9e24ddaaaaa3
```

- Pour API plus de détails, consultez la section CreateQueryLoggingConfigRéférence des AWS Tools for PowerShell applets de commande.

### `New-R53ReusableDelegationSet`

L'exemple de code suivant montre comment utiliser`New-R53ReusableDelegationSet`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un ensemble de délégation réutilisable de 4 serveurs de noms qui peuvent être réutilisés par plusieurs zones hébergées.

```
New-R53ReusableDelegationSet -CallerReference testcallerreference
```

Sortie :

```
DelegationSet                      Location
-------------                      --------
Amazon.Route53.Model.DelegationSet https://route53.amazonaws.com/2013-04-01/
delegationset/N23DS9XAAAAAXM
```

- Pour API plus de détails, consultez la section CreateReusableDelegationSetRéférence des AWS Tools for PowerShell applets de commande.

### `Register-R53VPCWithHostedZone`

L'exemple de code suivant montre comment utiliser`Register-R53VPCWithHostedZone`.

Outils pour PowerShell

Exemple 1 : Cet exemple associe la zone spécifiée VPC à la zone hébergée privée.

```
Register-R53VPCWithHostedZone -HostedZoneId Z3MEQ8T7HAAAAF -VPC_VPCId vpc-f1b9aaaa -
VPC_VPCRegion us-east-1
```

Sortie :

```
Id                    Status  SubmittedAt         Comment
--                    ------  -----------         -------
/change/C3SCAAA633Z6DX PENDING 01/28/2020 19:32:02
```

- Pour API plus de détails, reportez-vous à la section A du manuel ssociateVPCWith HostedZone de référence des AWS Tools for PowerShell applets de commande.

### `Remove-R53HostedZone`

L'exemple de code suivant montre comment utiliser`Remove-R53HostedZone`.

Outils pour PowerShell

Exemple 1 : Supprime la zone hébergée avec l'ID spécifié. Vous serez invité à confirmer avant que la commande ne soit exécutée, sauf si vous ajoutez le paramètre de commutation -Force.

```
Remove-R53HostedZone -Id Z1PA6795UKMFR9
```

- Pour API plus de détails, consultez la section DeleteHostedZoneRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-R53QueryLoggingConfig`

L'exemple de code suivant montre comment utiliser`Remove-R53QueryLoggingConfig`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la configuration spécifiée pour la journalisation des DNS requêtes.

```
Remove-R53QueryLoggingConfig -Id ee528e95-4e03-4fdc-9d28-9e24daaa20063
```

- Pour API plus de détails, consultez la section DeleteQueryLoggingConfigRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-R53ReusableDelegationSet`

L'exemple de code suivant montre comment utiliser`Remove-R53ReusableDelegationSet`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le jeu de délégation réutilisable spécifié.

```
Remove-R53ReusableDelegationSet -Id N23DS9X4AYAAAM
```

- Pour API plus de détails, consultez la section DeleteReusableDelegationSetRéférence des AWS Tools for PowerShell applets de commande.

### `Unregister-R53VPCFromHostedZone`

L'exemple de code suivant montre comment utiliser`Unregister-R53VPCFromHostedZone`.

Outils pour PowerShell

Exemple 1 : Cet exemple dissocie la zone spécifiée VPC de la zone hébergée privée.

```
Unregister-R53VPCFromHostedZone -HostedZoneId Z3MEQ8T7HAAAAF -VPC_VPCId vpc-f1b9aaaa
 -VPC_VPCRegion us-east-1
```

Sortie :

```
Id                    Status  SubmittedAt         Comment
--                    ------  -----------         -------
/change/C2XFCAAAA9HKZG PENDING 01/28/2020 10:35:55
```

- Pour API plus de détails, voir D isassociateVPCFrom HostedZone dans la référence des AWS Tools for PowerShell applets de commande.

### `Update-R53HostedZoneComment`

L'exemple de code suivant montre comment utiliser`Update-R53HostedZoneComment`.

Outils pour PowerShell

Exemple 1 : Cette commande met à jour le commentaire pour la zone hébergée spécifiée.

```
Update-R53HostedZoneComment -Id Z385PDS6AAAAAR -Comment "This is my first hosted
 zone"
```

Sortie :

```
Id                    : /hostedzone/Z385PDS6AAAAAR
Name                  : example.com.
CallerReference       : C5B55555-7147-EF04-8341-69131E805C89
Config                : Amazon.Route53.Model.HostedZoneConfig
ResourceRecordSetCount : 9
LinkedService         :
```

- Pour API plus de détails, consultez la section UpdateHostedZoneCommentRéférence des AWS Tools for PowerShell applets de commande.

# Exemples d'Amazon S3 utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS Tools for PowerShell aide d'Amazon S3.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `Copy-S3Object`

L'exemple de code suivant montre comment utiliser`Copy-S3Object`.

Outils pour PowerShell

Exemple 1 : Cette commande copie l'objet « sample.txt » du bucket « test-files » vers le même bucket mais avec une nouvelle clé « sample-copy.txt ».

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -DestinationKey
 sample-copy.txt
```

Exemple 2 : Cette commande copie l'objet « sample.txt » du bucket « test-files » vers le bucket « backup-files » avec la clé « sample-copy.txt ».

```
Copy-S3Object -BucketName amzn-s3-demo-source-bucket -Key sample.txt -DestinationKey
 sample-copy.txt -DestinationBucket amzn-s3-demo-destination-bucket
```

Exemple 3 : Cette commande télécharge l'objet « sample.txt » du bucket « test-files » vers un fichier local nommé « local-sample.txt ».

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-
sample.txt
```

Exemple 4 : télécharge l'objet unique dans le fichier spécifié. Le fichier téléchargé se trouve dans c:\downloads\data\archive.zip

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key data/archive.zip -LocalFolder c:
\downloads
```

Exemple 5 : télécharge tous les objets correspondant au préfixe de clé spécifié dans le dossier local. La hiérarchie des clés relative sera préservée sous forme de sous-dossiers dans l'emplacement de téléchargement global.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data -LocalFolder c:
\downloads
```

- Pour API plus de détails, consultez la section CopyObjectRéférence des AWS Tools for PowerShell applets de commande.

### `Get-S3ACL`

L'exemple de code suivant montre comment utiliser`Get-S3ACL`.

Outils pour PowerShell

Exemple 1 : La commande obtient les détails du propriétaire de l'objet S3.

```
Get-S3ACL -BucketName 'amzn-s3-demo-bucket' -key 'initialize.ps1' -Select
 AccessControlList.Owner
```

Sortie :

```
DisplayName Id
----------- --
testusername      9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544
```

- Pour API plus de détails, consultez la section Référence de ACL l'AWS Tools for PowerShell applet de commande Get in.

### `Get-S3Bucket`

L'exemple de code suivant montre comment utiliser`Get-S3Bucket`.

Outils pour PowerShell

Exemple 1 : Cette commande renvoie tous les compartiments S3.

```
Get-S3Bucket
```

Exemple 2 : Cette commande renvoie un bucket nommé « test-files »

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- Pour API plus de détails, consultez la section ListBucketsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-S3BucketAccelerateConfiguration`

L'exemple de code suivant montre comment utiliser`Get-S3BucketAccelerateConfiguration`.

Outils pour PowerShell

Exemple 1 : Cette commande renvoie la valeur Enabled, si les paramètres d'accélération du transfert sont activés pour le compartiment spécifié.

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

Sortie :

```
Value
-----
Enabled
```

- Pour API plus de détails, consultez la section GetBucketAccelerateConfigurationRéférence des AWS Tools for PowerShell applets de commande.

### `Get-S3BucketAnalyticsConfiguration`

L'exemple de code suivant montre comment utiliser`Get-S3BucketAnalyticsConfiguration`.

Outils pour PowerShell

Exemple 1 : Cette commande renvoie les détails du filtre d'analyse nommé « testfilter » dans le compartiment S3 donné.

```
Get-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId
 'testfilter'
```

- Pour API plus de détails, consultez la section GetBucketAnalyticsConfigurationRéférence des AWS Tools for PowerShell applets de commande.

### `Get-S3BucketAnalyticsConfigurationList`

L'exemple de code suivant montre comment utiliser`Get-S3BucketAnalyticsConfigurationList`.

Outils pour PowerShell

Exemple 1 : Cette commande renvoie les 100 premières configurations d'analyse du compartiment S3 donné.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Pour API plus de détails, consultez la section ListBucketAnalyticsConfigurationsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-S3BucketEncryption`

L'exemple de code suivant montre comment utiliser`Get-S3BucketEncryption`.

Outils pour PowerShell

Exemple 1 : Cette commande renvoie toutes les règles de chiffrement côté serveur associées au bucket donné.

```
Get-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

- Pour API plus de détails, consultez la section GetBucketEncryptionRéférence des AWS Tools for PowerShell applets de commande.

### `Get-S3BucketInventoryConfiguration`

L'exemple de code suivant montre comment utiliser`Get-S3BucketInventoryConfiguration`.

Outils pour PowerShell

Exemple 1 : Cette commande renvoie les détails de l'inventaire nommé « testinventory » pour le compartiment S3 donné.

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId
 'testinventory'
```

- Pour API plus de détails, consultez la section GetBucketInventoryConfigurationRéférence des AWS Tools for PowerShell applets de commande.

### `Get-S3BucketInventoryConfigurationList`

L'exemple de code suivant montre comment utiliser`Get-S3BucketInventoryConfigurationList`.

Outils pour PowerShell

Exemple 1 : Cette commande renvoie les 100 premières configurations d'inventaire du compartiment S3 donné.

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Pour API plus de détails, consultez la section ListBucketInventoryConfigurationsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-S3BucketLocation`

L'exemple de code suivant montre comment utiliser`Get-S3BucketLocation`.

Outils pour PowerShell

Exemple 1 : Cette commande renvoie la contrainte d'emplacement pour le bucket « s3testbucket », s'il existe une contrainte.

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

Sortie :

```
Value
-----
ap-south-1
```

- Pour API plus de détails, consultez la section GetBucketLocationRéférence des AWS Tools for PowerShell applets de commande.

### `Get-S3BucketLogging`

L'exemple de code suivant montre comment utiliser`Get-S3BucketLogging`.

Outils pour PowerShell

Exemple 1 : Cette commande renvoie l'état de journalisation pour le compartiment spécifié.

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

Sortie :

```
TargetBucketName   Grants TargetPrefix
----------------   ------ ------------
testbucket1        {}     testprefix
```

- Pour API plus de détails, consultez la section GetBucketLoggingRéférence des AWS Tools for PowerShell applets de commande.

### `Get-S3BucketMetricsConfiguration`

L'exemple de code suivant montre comment utiliser`Get-S3BucketMetricsConfiguration`.

Outils pour PowerShell

Exemple 1 : Cette commande renvoie les détails du filtre de métriques nommé « testfilter » pour le compartiment S3 donné.

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId
 'testfilter'
```

- Pour API plus de détails, consultez la section GetBucketMetricsConfigurationRéférence des AWS Tools for PowerShell applets de commande.

### `Get-S3BucketNotification`

L'exemple de code suivant montre comment utiliser`Get-S3BucketNotification`.

Outils pour PowerShell

Exemple 1 : Cet exemple récupère la configuration des notifications du bucket donné

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty
 TopicConfigurations
```

Sortie :

```
Id   Topic
--   -----
mimo arn:aws:sns:eu-west-1:123456789012:topic-1
```

- Pour API plus de détails, consultez la section GetBucketNotificationRéférence des AWS Tools for PowerShell applets de commande.

### `Get-S3BucketPolicy`

L'exemple de code suivant montre comment utiliser`Get-S3BucketPolicy`.

Outils pour PowerShell

Exemple 1 : Cette commande génère la politique de compartiment associée au compartiment S3 donné.

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Pour API plus de détails, consultez la section GetBucketPolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Get-S3BucketPolicyStatus`

L'exemple de code suivant montre comment utiliser`Get-S3BucketPolicyStatus`.

Outils pour PowerShell

Exemple 1 : Cette commande renvoie l'état de la politique pour le compartiment S3 donné, indiquant si le compartiment est public.

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- Pour API plus de détails, consultez la section GetBucketPolicyStatusRéférence des AWS Tools for PowerShell applets de commande.

### `Get-S3BucketReplication`

L'exemple de code suivant montre comment utiliser`Get-S3BucketReplication`.

Outils pour PowerShell

Exemple 1 : renvoie les informations de configuration de réplication définies sur le compartiment nommé « mybucket ».

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Pour API plus de détails, consultez la section GetBucketReplicationRéférence des AWS Tools for PowerShell applets de commande.

### `Get-S3BucketRequestPayment`

L'exemple de code suivant montre comment utiliser`Get-S3BucketRequestPayment`.

Outils pour PowerShell

Exemple 1 : renvoie la configuration de paiement de la demande pour le compartiment nommé « mybucket ». Par défaut, le propriétaire du bucket paie les téléchargements depuis le bucket.

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- Pour API plus de détails, consultez la section GetBucketRequestPaymentRéférence des AWS Tools for PowerShell applets de commande.

### `Get-S3BucketTagging`

L'exemple de code suivant montre comment utiliser`Get-S3BucketTagging`.

Outils pour PowerShell

Exemple 1 : Cette commande renvoie toutes les balises associées au bucket donné.

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- Pour API plus de détails, consultez la section GetBucketTaggingRéférence des AWS Tools for PowerShell applets de commande.

### `Get-S3BucketVersioning`

L'exemple de code suivant montre comment utiliser`Get-S3BucketVersioning`.

Outils pour PowerShell

Exemple 1 : Cette commande renvoie l'état du versionnement par rapport au bucket donné.

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- Pour API plus de détails, consultez la section GetBucketVersioningRéférence des AWS Tools for PowerShell applets de commande.

### `Get-S3BucketWebsite`

L'exemple de code suivant montre comment utiliser`Get-S3BucketWebsite`.

Outils pour PowerShell

Exemple 1 : Cette commande renvoie les détails des configurations de site Web statiques du compartiment S3 donné.

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- Pour API plus de détails, consultez la section GetBucketWebsiteRéférence des AWS Tools for PowerShell applets de commande.

### `Get-S3CORSConfiguration`

L'exemple de code suivant montre comment utiliser`Get-S3CORSConfiguration`.

Outils pour PowerShell

Exemple 1 : Cette commande renvoie un objet contenant toutes les règles CORS de configuration correspondant au compartiment S3 donné.

```
Get-S3CORSConfiguration -BucketName 'amzn-s3-demo-bucket' -Select
 Configuration.Rules
```

Sortie :

```
AllowedMethods : {PUT, POST, DELETE}
AllowedOrigins : {http://www.example1.com}
Id             :
ExposeHeaders  : {}
MaxAgeSeconds  : 0
AllowedHeaders : {*}

AllowedMethods : {PUT, POST, DELETE}
AllowedOrigins : {http://www.example2.com}
Id             :
ExposeHeaders  : {}
MaxAgeSeconds  : 0
AllowedHeaders : {*}

AllowedMethods : {GET}
AllowedOrigins : {*}
Id             :
ExposeHeaders  : {}
```

```
MaxAgeSeconds  : 0
AllowedHeaders : {}
```

- Pour API plus de détails, voir G etCORSConfiguration dans le manuel de référence des AWS Tools for PowerShell applets de commande.

### `Get-S3LifecycleConfiguration`

L'exemple de code suivant montre comment utiliser`Get-S3LifecycleConfiguration`.

Outils pour PowerShell

Exemple 1 : Cet exemple extrait la configuration du cycle de vie du compartiment.

```
Get-S3LifecycleConfiguration -BucketName amzn-s3-demo-bucket
```

Sortie :

```
Rules
-----
{Remove-in-150-days, Archive-to-Glacier-in-30-days}
```

- Pour API plus de détails, consultez la section GetLifecycleConfigurationRéférence des AWS Tools for PowerShell applets de commande.

### `Get-S3Object`

L'exemple de code suivant montre comment utiliser`Get-S3Object`.

Outils pour PowerShell

Exemple 1 : Cette commande récupère les informations relatives à tous les éléments du bucket « test-files ».

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

Exemple 2 : Cette commande récupère les informations relatives à l'élément « sample.txt » depuis le bucket « test-files ».

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Exemple 3 : Cette commande récupère les informations relatives à tous les éléments portant le préfixe « sample » à partir du bucket « test-files ».

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- Pour API plus de détails, consultez la section ListObjectsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-S3ObjectLockConfiguration`

L'exemple de code suivant montre comment utiliser`Get-S3ObjectLockConfiguration`.

Outils pour PowerShell

Exemple 1 : Cette commande renvoie la valeur « Enabled » si la configuration Object Lock est activée pour le compartiment S3 donné.

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select
 ObjectLockConfiguration.ObjectLockEnabled
```

Sortie :

```
Value
-----
Enabled
```

- Pour API plus de détails, consultez la section GetObjectLockConfigurationRéférence des AWS Tools for PowerShell applets de commande.

### `Get-S3ObjectMetadata`

L'exemple de code suivant montre comment utiliser`Get-S3ObjectMetadata`.

Outils pour PowerShell

Exemple 1 : Cette commande renvoie les métadonnées de l'objet avec la clé « ListTrusts .txt » dans le compartiment S3 donné.

```
Get-S3ObjectMetadata -BucketName 'amzn-s3-demo-bucket' -Key 'ListTrusts.txt'
```

Sortie :

```
Headers                                       : Amazon.S3.Model.HeadersCollection
Metadata                                      : Amazon.S3.Model.MetadataCollection
DeleteMarker                                  :
AcceptRanges                                  : bytes
ContentRange                                  :
Expiration                                    :
RestoreExpiration                             :
RestoreInProgress                             : False
LastModified                                  : 01/01/2020 08:02:05
ETag                                          : "d000011112a222e333e3bb4ee5d43d21"
MissingMeta                                   : 0
VersionId                                     : null
Expires                                       : 01/01/0001 00:00:00
WebsiteRedirectLocation                       :
ServerSideEncryptionMethod                    : AES256
ServerSideEncryptionCustomerMethod            :
ServerSideEncryptionKeyManagementServiceKeyId :
ReplicationStatus                             :
PartsCount                                    :
ObjectLockLegalHoldStatus                     :
ObjectLockMode                                :
ObjectLockRetainUntilDate                     : 01/01/0001 00:00:00
StorageClass                                  :
RequestCharged                                :
```

- Pour API plus de détails, consultez la section GetObjectMetadataRéférence des AWS Tools for PowerShell applets de commande.

### `Get-S3ObjectRetention`

L'exemple de code suivant montre comment utiliser`Get-S3ObjectRetention`.

Outils pour PowerShell

Exemple 1 : La commande renvoie le mode et la date jusqu'à ce que l'objet soit conservé.

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- Pour API plus de détails, consultez la section GetObjectRetentionRéférence des AWS Tools for PowerShell applets de commande.

### `Get-S3ObjectTagSet`

L'exemple de code suivant montre comment utiliser`Get-S3ObjectTagSet`.

Outils pour PowerShell

Exemple 1 : L'exemple renvoie les balises associées à l'objet présent dans le compartiment S3 donné.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

Sortie :

```
Key  Value
---  -----
test value
```

- Pour API plus de détails, consultez la section GetObjectTaggingRéférence des AWS Tools for PowerShell applets de commande.

### `Get-S3PreSignedURL`

L'exemple de code suivant montre comment utiliser`Get-S3PreSignedURL`.

Outils pour PowerShell

Exemple 1 : La commande renvoie une signature préalable URL pour une clé spécifiée et une date d'expiration.

```
Get-S3PreSignedURL -BucketName 'amzn-s3-demo-bucket' -Key 'testkey' -Expires
 '2023-11-16'
```

Exemple 2 : La commande renvoie une signature préalable URL pour un bucket de répertoire avec une clé spécifiée et une date d'expiration.

```
[Amazon.AWSConfigsS3]::UseSignatureVersion4 = $true
      Get-S3PreSignedURL -BucketName amzn-s3-demo-bucket--usw2-az1--x-s3 -Key
 'testkey' -Expire '2023-11-17'
```

- Pour API plus de détails, consultez la section GetPreSignedURLRéférence des AWS Tools for PowerShell applets de commande.

### `Get-S3PublicAccessBlock`

L'exemple de code suivant montre comment utiliser`Get-S3PublicAccessBlock`.

Outils pour PowerShell

Exemple 1 : La commande renvoie la configuration du bloc d'accès public du compartiment S3 donné.

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- Pour API plus de détails, consultez la section GetPublicAccessBlockRéférence des AWS Tools for PowerShell applets de commande.

### `Get-S3Version`

L'exemple de code suivant montre comment utiliser`Get-S3Version`.

Outils pour PowerShell

Exemple 1 : Cette commande renvoie les métadonnées relatives à toutes les versions des objets du compartiment S3 donné.

```
Get-S3Version -BucketName 'amzn-s3-demo-bucket'
```

Sortie :

```
IsTruncated         : False
KeyMarker           :
VersionIdMarker     :
NextKeyMarker       :
NextVersionIdMarker :
Versions            : {EC2.txt, EC2MicrosoftWindowsGuide.txt, ListDirectories.json,
 ListTrusts.json}
Name                : s3testbucket
Prefix              :
MaxKeys             : 1000
CommonPrefixes      : {}
```

```
Delimiter           :
```

- Pour API plus de détails, consultez la section ListVersionsRéférence des AWS Tools for PowerShell applets de commande.

### `New-S3Bucket`

L'exemple de code suivant montre comment utiliser`New-S3Bucket`.

Outils pour PowerShell

Exemple 1 : Cette commande crée un nouveau bucket privé nommé « sample-bucket ».

```
New-S3Bucket -BucketName amzn-s3-demo-bucket
```

Exemple 2 : Cette commande crée un nouveau compartiment nommé « sample-bucket » avec des autorisations de lecture-écriture.

```
New-S3Bucket -BucketName amzn-s3-demo-bucket -PublicReadWrite
```

Exemple 3 : Cette commande crée un nouveau compartiment nommé « sample-bucket » avec des autorisations en lecture seule.

```
New-S3Bucket -BucketName amzn-s3-demo-bucket -PublicReadOnly
```

Exemple 4 : Cette commande crée un nouveau compartiment de répertoire nommé « samplebucket--use1-az5--x-s3 » avec. PutBucketConfiguration

```
$bucketConfiguration = @{
      BucketInfo = @{
          DataRedundancy = 'SingleAvailabilityZone'
          Type = 'Directory'
      }
      Location = @{
          Name = 'usw2-az1'
          Type = 'AvailabilityZone'
      }
    }
New-S3Bucket -BucketName amzn-s3-demo-bucket--usw2-az1--x-s3 -BucketConfiguration
 $bucketConfiguration -Region us-west-2
```

- Pour API plus de détails, consultez la section PutBucketRéférence des AWS Tools for PowerShell applets de commande.

### `Read-S3Object`

L'exemple de code suivant montre comment utiliser`Read-S3Object`.

Outils pour PowerShell

Exemple 1 : Cette commande récupère l'élément « sample.txt » du bucket « test-files » et l'enregistre dans un fichier nommé « local-sample.txt » à l'emplacement actuel. Il n'est pas nécessaire que le fichier « local-sample.txt » existe pour que cette commande soit appelée.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt
```

Exemple 2 : Cette commande extrait le répertoire virtuel « DIR » du bucket « test-files » et l'enregistre dans un dossier nommé « Local- DIR » à l'emplacement actuel. Le dossier « Local-DIR » n'a pas besoin d'exister pour que cette commande soit appelée.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

Exemple 3 : télécharge tous les objets dont les clés se terminent par « .json » depuis les compartiments dont le nom contient « config » vers les fichiers du dossier spécifié. Les clés d'objet sont utilisées pour définir les noms de fichiers.

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -
like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

- Pour API plus de détails, consultez la section GetObjectRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-S3Bucket`

L'exemple de code suivant montre comment utiliser`Remove-S3Bucket`.

Outils pour PowerShell

Exemple 1 : Cette commande supprime tous les objets et toutes les versions d'objets du bucket « test-files », puis supprime le bucket. La commande vous demandera une confirmation avant

de continuer. Ajoutez le commutateur -Force pour supprimer la confirmation. Notez que les compartiments qui ne sont pas vides ne peuvent pas être supprimés.

```
Remove-S3Bucket -BucketName amzn-s3-demo-bucket -DeleteBucketContent
```

- Pour API plus de détails, consultez la section DeleteBucketRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-S3BucketAnalyticsConfiguration`

L'exemple de code suivant montre comment utiliser`Remove-S3BucketAnalyticsConfiguration`.

Outils pour PowerShell

Exemple 1 : La commande supprime le filtre d'analyse nommé « testfilter » dans le compartiment S3 donné.

```
Remove-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId
 'testfilter'
```

- Pour API plus de détails, consultez la section DeleteBucketAnalyticsConfigurationRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-S3BucketEncryption`

L'exemple de code suivant montre comment utiliser`Remove-S3BucketEncryption`.

Outils pour PowerShell

Exemple 1 : Cela désactive le chiffrement activé pour le compartiment S3 fourni.

```
Remove-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketEncryption (DeleteBucketEncryption)" on
 target "s3casetestbucket".
```

```
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Pour API plus de détails, consultez la section DeleteBucketEncryptionRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-S3BucketInventoryConfiguration`

L'exemple de code suivant montre comment utiliser`Remove-S3BucketInventoryConfiguration`.

Outils pour PowerShell

Exemple 1 : Cette commande supprime l'inventaire nommé « testInventoryName » correspondant au compartiment S3 donné.

```
Remove-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId
 'testInventoryName'
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketInventoryConfiguration
 (DeleteBucketInventoryConfiguration)" on target "s3testbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Pour API plus de détails, consultez la section DeleteBucketInventoryConfigurationRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-S3BucketMetricsConfiguration`

L'exemple de code suivant montre comment utiliser`Remove-S3BucketMetricsConfiguration`.

Outils pour PowerShell

Exemple 1 : La commande supprime le filtre de métriques nommé « testmetrics » dans le compartiment S3 donné.

```
Remove-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId
 'testmetrics'
```

- Pour API plus de détails, consultez la section DeleteBucketMetricsConfigurationRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-S3BucketPolicy`

L'exemple de code suivant montre comment utiliser`Remove-S3BucketPolicy`.

Outils pour PowerShell

Exemple 1 : La commande supprime la politique de compartiment associée au compartiment S3 donné.

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Pour API plus de détails, consultez la section DeleteBucketPolicyRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-S3BucketReplication`

L'exemple de code suivant montre comment utiliser`Remove-S3BucketReplication`.

Outils pour PowerShell

Exemple 1 : Supprime la configuration de réplication associée au bucket nommé « mybucket ». Notez que cette opération nécessite une autorisation pour l'DeleteReplicationConfiguration action s3 :. Vous serez invité à confirmer avant que l'opération ne se poursuive. Pour supprimer la confirmation, utilisez le commutateur -Force.

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Pour API plus de détails, consultez la section DeleteBucketReplicationRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-S3BucketTagging`

L'exemple de code suivant montre comment utiliser`Remove-S3BucketTagging`.

Outils pour PowerShell

Exemple 1 : Cette commande supprime toutes les balises associées au compartiment S3 donné.

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketTagging (DeleteBucketTagging)" on target
 "s3testbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Pour API plus de détails, consultez la section DeleteBucketTaggingRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-S3BucketWebsite`

L'exemple de code suivant montre comment utiliser`Remove-S3BucketWebsite`.

Outils pour PowerShell

Exemple 1 : Cette commande désactive la propriété d'hébergement statique du site Web du compartiment S3 donné.

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketWebsite (DeleteBucketWebsite)" on target
 "s3testbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Pour API plus de détails, consultez la section DeleteBucketWebsiteRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-S3CORSConfiguration`

L'exemple de code suivant montre comment utiliser`Remove-S3CORSConfiguration`.

Outils pour PowerShell

Exemple 1 : Cette commande supprime la CORS configuration du compartiment S3 donné.

```
Remove-S3CORSConfiguration -BucketName 'amzn-s3-demo-bucket'
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3CORSConfiguration (DeleteCORSConfiguration)" on
 target "s3testbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Pour API plus de détails, reportez-vous à la section D eleteCORSConfiguration dans la référence des AWS Tools for PowerShell applets de commande.

### `Remove-S3LifecycleConfiguration`

L'exemple de code suivant montre comment utiliser`Remove-S3LifecycleConfiguration`.

Outils pour PowerShell

Exemple 1 : La commande supprime toutes les règles de cycle de vie pour le compartiment S3 donné.

```
Remove-S3LifecycleConfiguration -BucketName 'amzn-s3-demo-bucket'
```

- Pour API plus de détails, consultez la section DeleteLifecycleConfigurationRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-S3MultipartUpload`

L'exemple de code suivant montre comment utiliser`Remove-S3MultipartUpload`.

Outils pour PowerShell

Exemple 1 : Cette commande annule les téléchargements partitionnés créés il y a moins de 5 jours.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -DaysBefore 5
```

Exemple 2 : Cette commande annule les téléchargements partitionnés créés avant le 2 janvier 2014.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate "Thursday,
 January 02, 2014"
```

Exemple 3 : Cette commande annule les téléchargements partitionnés créés avant le 2 janvier 2014, 10:45:37.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate "2014/01/02
 10:45:37"
```

- Pour API plus de détails, consultez la section AbortMultipartUploadRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-S3Object`

L'exemple de code suivant montre comment utiliser`Remove-S3Object`.

Outils pour PowerShell

Exemple 1 : Cette commande supprime l'objet « sample.txt » du bucket « test-files ». Vous êtes invité à confirmer avant l'exécution de la commande ; pour supprimer l'invite, utilisez le commutateur -Force.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Exemple 2 : Cette commande supprime la version spécifiée de l'objet « sample.txt » du bucket « test-files », en supposant que le bucket a été configuré pour activer les versions de l'objet.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId
 HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

Exemple 3 : cette commande supprime les objets « sample1.txt », « sample2.txt » et « sample3.txt » du bucket « test-files » en une seule opération par lots. La réponse du service listera toutes les clés traitées, quel que soit le statut de réussite ou d'erreur de la suppression. Pour obtenir uniquement les erreurs pour les clés qui n'ont pas pu être traitées par le service, ajoutez le ReportErrorsOnly paramètre - (ce paramètre peut également être spécifié avec l'alias -Quiet).

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",
 "sample2.txt", "sample3.txt" )
```

Exemple 4 : Cet exemple utilise une expression en ligne avec le KeyCollection paramètre - pour obtenir les clés des objets à supprimer. Get-S3Objectrenvoie une collection d'instances Amazon.S3.Model.S3Object, dont chacune possède un membre clé de type chaîne identifiant l'objet.

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object
 "test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

Exemple 5 : Cet exemple obtient tous les objets dont le préfixe clé est « préfixe/sous-préfixe » dans le compartiment et les supprime. Notez que les objets entrants sont traités un par un. Pour les collections volumineuses, pensez à transmettre la collection au paramètre - InputObject (alias -S3ObjectCollection) de l'applet de commande pour permettre à la suppression de se produire par lots avec un seul appel au service.

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |
 Remove-S3Object -Force
```

Exemple 6 : Cet exemple dirige une collection d'ObjectVersion instances Amazon.S3.Model.S3 qui représentent des marqueurs de suppression vers l'applet de commande pour suppression. Notez que les objets entrants sont traités un par un. Pour les collections volumineuses, pensez à transmettre la collection au paramètre - InputObject (alias -S3ObjectCollection) de l'applet de commande pour permettre à la suppression de se produire par lots avec un seul appel au service.

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where
 {$_.IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

Exemple 7 : Ce script montre comment supprimer par lots un ensemble d'objets (dans ce cas, des marqueurs de suppression) en créant un tableau d'objets à utiliser avec le KeyAndVersionCollection paramètre -.

```
$keyVersions = @()
$markers = (Get-S3Version -BucketName $BucketName).Versions | Where
 {$_.IsDeleteMarker -eq "True"}
foreach ($marker in $markers) { $keyVersions += @{ Key = $marker.Key; VersionId =
 $marker.VersionId } }
Remove-S3Object -BucketName $BucketName -KeyAndVersionCollection $keyVersions -Force
```

- Pour API plus de détails, consultez la section DeleteObjectsRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-S3ObjectTagSet`

L'exemple de code suivant montre comment utiliser`Remove-S3ObjectTagSet`.

Outils pour PowerShell

Exemple 1 : Cette commande supprime toutes les balises associées à l'objet avec la clé « testfile.txt » dans le compartiment S3 donné.

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -Select
 '^Key'
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3ObjectTagSet (DeleteObjectTagging)" on target
 "testfile.txt".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
testfile.txt
```

- Pour API plus de détails, consultez la section DeleteObjectTaggingRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-S3PublicAccessBlock`

L'exemple de code suivant montre comment utiliser`Remove-S3PublicAccessBlock`.

Outils pour PowerShell

Exemple 1 : Cette commande désactive le paramètre de blocage de l'accès public pour le bucket donné.

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select
 '^BucketName'
```

Sortie :

```
s3testbucket
```

- Pour API plus de détails, consultez la section DeletePublicAccessBlockRéférence des AWS Tools for PowerShell applets de commande.

### `Set-S3BucketEncryption`

L'exemple de code suivant montre comment utiliser`Set-S3BucketEncryption`.

Outils pour PowerShell

Exemple 1 : Cette commande active le chiffrement côté AES256 serveur par défaut avec les clés gérées Amazon S3 (SSE-S3) sur le compartiment donné.

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =
 @{ServerSideEncryptionAlgorithm = "AES256"}}
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- Pour API plus de détails, consultez la section PutBucketEncryptionRéférence des AWS Tools for PowerShell applets de commande.

### `Test-S3Bucket`

L'exemple de code suivant montre comment utiliser`Test-S3Bucket`.

Outils pour PowerShell

Exemple 1 : cette commande renvoie True si le bucket existe, False dans le cas contraire. La commande renvoie True même si le bucket n'appartient pas à l'utilisateur.

```
Test-S3Bucket -BucketName amzn-s3-demo-bucket
```

- Pour API plus de détails, consultez la section Test-S3BucketRéférence des AWS Tools for PowerShell applets de commande.

### `Write-S3BucketAccelerateConfiguration`

L'exemple de code suivant montre comment utiliser`Write-S3BucketAccelerateConfiguration`.

Outils pour PowerShell

Exemple 1 : Cette commande active l'accélération du transfert pour le compartiment S3 donné.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -
AccelerateConfiguration_Status $statusVal
```

- Pour API plus de détails, consultez la section PutBucketAccelerateConfigurationRéférence des AWS Tools for PowerShell applets de commande.

### `Write-S3BucketNotification`

L'exemple de code suivant montre comment utiliser`Write-S3BucketNotification`.

Outils pour PowerShell

Exemple 1 : Cet exemple configure la configuration des SNS rubriques pour l'événement S3 ObjectRemovedDelete et active les notifications pour le compartiment s3 donné

```
$topic =  [Amazon.S3.Model.TopicConfiguration] @{
  Id = "delete-event"
  Topic = "arn:aws:sns:eu-west-1:123456789012:topic-1"
  Event = [Amazon.S3.EventType]::ObjectRemovedDelete
```

```
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -TopicConfiguration
 $topic
```

Exemple 2 : Cet exemple active les notifications ObjectCreatedAll pour le bucket donné qui l'envoient à la fonction Lambda.

```
$lambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = "s3:ObjectCreated:*"
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:rdplock"
  Id = "ObjectCreated-Lambda"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".pem"}
      )
    }
  }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $lambdaConfig
```

Exemple 3 : Cet exemple crée 2 configurations Lambda différentes sur la base d'un suffixe clé différent et configure les deux en une seule commande.

```
#Lambda Config 1

$firstLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = "s3:ObjectCreated:*"
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifynet"
  Id = "ObjectCreated-dada-ps1"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".ps1"}
      )
    }
  }
```

```
}

#Lambda Config 2

$secondlambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = [Amazon.S3.EventType]::ObjectCreatedAll
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifyssm"
  Id = "ObjectCreated-dada-json"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".json"}
      )
    }
  }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $firstLambdaConfig,$secondlambdaConfig
```

- Pour API plus de détails, consultez la section PutBucketNotificationRéférence des AWS Tools for PowerShell applets de commande.

### `Write-S3BucketReplication`

L'exemple de code suivant montre comment utiliser`Write-S3BucketReplication`.

Outils pour PowerShell

Exemple 1 : Cet exemple définit une configuration de réplication avec une règle unique permettant de répliquer dans le compartiment « exampletargetbucket » tous les nouveaux objets créés avec le préfixe de nom clé « » dans le compartiment « examplebucket »TaxDocs.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
```

```
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

Exemple 2 : Cet exemple définit une configuration de réplication avec plusieurs règles permettant de répliquer dans le compartiment « exampletargetbucket » tous les nouveaux objets créés avec le préfixe de nom de clé « » ou « ». TaxDocs OtherDocs Les préfixes clés ne doivent pas se chevaucher.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-bucket" }

$rule2 = New-Object Amazon.S3.Model.ReplicationRule
$rule2.ID = "Rule-2"
$rule2.Status = "Enabled"
$rule2.Prefix = "OtherDocs"
$rule2.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1,$rule2
}

Write-S3BucketReplication @params
```

Exemple 3 : Cet exemple met à jour la configuration de réplication sur le compartiment spécifié afin de désactiver la règle contrôlant la réplication des objets portant le préfixe de nom clé « » vers le compartiment TaxDocs « exampletargetbucket ».

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Disabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-bucket" }
```

```
$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

- Pour API plus de détails, consultez la section PutBucketReplicationRéférence des AWS Tools for PowerShell applets de commande.

### `Write-S3BucketRequestPayment`

L'exemple de code suivant montre comment utiliser`Write-S3BucketRequestPayment`.

Outils pour PowerShell

Exemple 1 : met à jour la configuration du paiement des demandes pour le compartiment nommé « mybucket » afin que le téléchargement soit facturé à la personne demandant des téléchargements depuis le compartiment. Par défaut, le propriétaire du bucket paie pour les téléchargements. Pour rétablir la valeur par défaut du paiement de la demande, utilisez « BucketOwner » pour le paramètre RequestPaymentConfiguration _Payer.

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -
RequestPaymentConfiguration_Payer Requester
```

- Pour API plus de détails, consultez la section PutBucketRequestPaymentRéférence des AWS Tools for PowerShell applets de commande.

### `Write-S3BucketTagging`

L'exemple de code suivant montre comment utiliser`Write-S3BucketTagging`.

Outils pour PowerShell

Exemple 1 : Cette commande applique deux balises à un compartiment nommé **`cloudtrail-test-2018`** : une balise avec une clé Stage et une valeur Test, et une balise avec une clé Environment et une valeur Alpha. Pour vérifier que les balises ont été ajoutées au compartiment,

exécutez**`Get-S3BucketTagging -BucketName bucket_name`**. Les résultats doivent indiquer les balises que vous avez appliquées au bucket lors de la première commande. Notez que cela **`Write-S3BucketTagging`** remplace l'ensemble de balises existant sur un bucket. Pour ajouter ou supprimer des balises individuelles, exécutez les API applets de commande Resource Groups et Tagging, et. **`Add-RGTResourceTag`** **`Remove-RGTResourceTag`** Vous pouvez également utiliser l'éditeur de balises dans la console AWS de gestion pour gérer les balises de compartiment S3.

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage";
 Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

Exemple 2 : Cette commande dirige un bucket nommé **`cloudtrail-test-2018`** vers l'**`Write-S3BucketTagging`**applet de commande. Il applique les balises Stage:Production et Department:Finance au bucket. Notez que cela **`Write-S3BucketTagging`** remplace l'ensemble de balises existant sur un bucket.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging -TagSet
 @( @{ Key="Stage"; Value="Production" }, @{ Key="Department"; Value="Finance" } )
```

- Pour API plus de détails, consultez la section PutBucketTaggingRéférence des AWS Tools for PowerShell applets de commande.

### `Write-S3BucketVersioning`

L'exemple de code suivant montre comment utiliser`Write-S3BucketVersioning`.

Outils pour PowerShell

Exemple 1 : La commande active la gestion des versions pour le compartiment S3 donné.

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -VersioningConfig_Status
 Enabled
```

- Pour API plus de détails, consultez la section PutBucketVersioningRéférence des AWS Tools for PowerShell applets de commande.

### `Write-S3BucketWebsite`

L'exemple de code suivant montre comment utiliser`Write-S3BucketWebsite`.

Outils pour PowerShell

Exemple 1 : La commande active l'hébergement du site Web pour le compartiment donné avec le document d'index « index.html » et le document d'erreur « error.html ».

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
 -WebsiteConfiguration_IndexDocumentSuffix 'index.html' -
WebsiteConfiguration_ErrorDocument 'error.html'
```

- Pour API plus de détails, consultez la section PutBucketWebsiteRéférence des AWS Tools for PowerShell applets de commande.

### `Write-S3LifecycleConfiguration`

L'exemple de code suivant montre comment utiliser`Write-S3LifecycleConfiguration`.

Outils pour PowerShell

Exemple 1 : Cet exemple écrit/remplace la configuration fournie dans le $NewRule. Cette configuration garantit de limiter les objets de portée avec des valeurs de préfixe et de balise données.

```
$NewRule = [Amazon.S3.Model.LifecycleRule] @{
  Expiration =  @{
   Days=  50
  }
  Id =  "Test-From-Write-cmdlet-1"
  Filter=  @{
   LifecycleFilterPredicate =  [Amazon.S3.Model.LifecycleAndOperator]@{
    Operands=  @(
     [Amazon.S3.Model.LifecyclePrefixPredicate] @{
      "Prefix" =  "py"
     },
     [Amazon.S3.Model.LifecycleTagPredicate] @{
      "Tag"=  @{
       "Key" =  "non-use"
       "Value" = "yes"
      }
     }
    )
   }
```

```
 }
 "Status"= 'Enabled'
 NoncurrentVersionExpiration = @{
  NoncurrentDays = 75
 }
}

Write-S3LifecycleConfiguration -BucketName amzn-s3-demo-bucket -Configuration_Rule
$NewRule
```

Exemple 2 : Cet exemple définit plusieurs règles avec le filtrage. $ ArchiveRule définit les objets à archiver dans 30 jours dans Glacier et 120 dans DeepArchive. $ ExpireRule expire les versions actuelles et précédentes en 150 jours pour les objets dont le préfixe « py » et le tag:key « archivé » sont définis sur « yes ».

```
$ExpireRule = [Amazon.S3.Model.LifecycleRule] @{
  Expiration =  @{
   Days=  150
  }
  Id =  "Remove-in-150-days"
  Filter=  @{
   LifecycleFilterPredicate =  [Amazon.S3.Model.LifecycleAndOperator]@{
    Operands=  @(
     [Amazon.S3.Model.LifecyclePrefixPredicate] @{
      "Prefix" =  "py"
     },
     [Amazon.S3.Model.LifecycleTagPredicate] @{
      "Tag"=  @{
       "Key" =  "archived"
       "Value" = "yes"
      }
     }
    )
   }
  }
  Status= 'Enabled'
  NoncurrentVersionExpiration = @{
   NoncurrentDays = 150
  }
 }

 $ArchiveRule = [Amazon.S3.Model.LifecycleRule] @{
  Expiration =  $null
```

```
 Id =  "Archive-to-Glacier-in-30-days"
 Filter=  @{
  LifecycleFilterPredicate =  [Amazon.S3.Model.LifecycleAndOperator]@{
   Operands= @(
    [Amazon.S3.Model.LifecyclePrefixPredicate] @{
     "Prefix" =  "py"
    },
    [Amazon.S3.Model.LifecycleTagPredicate] @{
     "Tag"=  @{
      "Key" =  "reviewed"
      "Value" = "yes"
     }
    }
   )
  }
 }
 Status = 'Enabled'
 NoncurrentVersionExpiration = @{
  NoncurrentDays = 75
 }
 Transitions = @(
  @{
   Days = 30
   "StorageClass"= 'Glacier'
  },
  @{
   Days = 120
   "StorageClass"= [Amazon.S3.S3StorageClass]::DeepArchive
  }
 )
}

Write-S3LifecycleConfiguration -BucketName amzn-s3-demo-bucket -Configuration_Rule
$ExpireRule,$ArchiveRule
```

- Pour API plus de détails, consultez la section PutLifecycleConfigurationRéférence des AWS Tools for PowerShell applets de commande.

### `Write-S3Object`

L'exemple de code suivant montre comment utiliser`Write-S3Object`.

Exemple 1 : cette commande télécharge le fichier unique « local-sample.txt » sur Amazon S3, créant un objet avec la clé « sample.txt » dans le compartiment « test-files ».

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-
sample.txt
```

Exemple 2 : cette commande télécharge le fichier unique « sample.txt » sur Amazon S3, créant un objet avec la clé « sample.txt » dans le compartiment « test-files ». Si le paramètre -Key n'est pas fourni, le nom du fichier est utilisé comme clé d'objet S3.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

Exemple 3 : cette commande télécharge le fichier unique « local-sample.txt » sur Amazon S3, créant un objet avec la clé « prefix/to/sample.txt » dans le compartiment « test-files ».

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -File .
\local-sample.txt
```

Exemple 4 : Cette commande télécharge tous les fichiers du sous-répertoire « Scripts » vers le bucket « test-files » et applique le préfixe de clé commun « » à chaque objet. SampleScripts Chaque fichier téléchargé aura une clé « SampleScripts /filename » où « filename » varie.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix
 SampleScripts\
```

Exemple 5 : Cette commande télécharge tous les fichiers*.ps1 du répertoire local « Scripts » vers le bucket « test-files » et applique le préfixe de clé commun « » à chaque objet. SampleScripts Chaque fichier téléchargé aura une clé « SampleScripts /filename.ps1 » où le « nom de fichier » varie.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix
 SampleScripts\ -SearchPattern *.ps1
```

Exemple 6 : Cette commande crée un nouvel objet S3 contenant la chaîne de contenu spécifiée avec la clé « sample.txt ».

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object
 contents"
```

Exemple 7 : Cette commande télécharge le fichier spécifié (le nom du fichier est utilisé comme clé) et applique les balises spécifiées au nouvel objet.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet
 @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

Exemple 8 : Cette commande télécharge de manière récursive le dossier spécifié et applique les balises spécifiées à tous les nouveaux objets.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles" -
Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- Pour API plus de détails, consultez la section PutObjectRéférence des AWS Tools for PowerShell applets de commande.

#### `Write-S3ObjectRetention`

L'exemple de code suivant montre comment utiliser`Write-S3ObjectRetention`.

Outils pour PowerShell

Exemple 1 : La commande active le mode de rétention de la gouvernance jusqu'à la date du 31 décembre 2019 00:00:00 pour l'objet « testfile.txt » dans le compartiment S3 donné.

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- Pour API plus de détails, consultez la section PutObjectRetentionRéférence des AWS Tools for PowerShell applets de commande.

## Exemples de S3 Glacier utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS Tools for PowerShell aide de S3 Glacier.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques



## Actions

### `Get-GLCJob`

L'exemple de code suivant montre comment utiliser`Get-GLCJob`.

Outils pour PowerShell

Exemple 1 : renvoie les détails de la tâche spécifiée. Lorsque la tâche est terminée avec succès, l'GCJobOutputapplet de commande Read- peut être utilisée pour récupérer le contenu de la tâche (une archive ou une liste d'inventaire) dans le système de fichiers local.

```
Get-GLCJob -VaultName myvault -JobId "op1x...JSbthM"
```

Sortie :

```
Action                       : ArchiveRetrieval
ArchiveId                    : o9O9j...X-TpIhQJw
ArchiveSHA256TreeHash        : 79f3ea754c02f58...dc57bf4395b
ArchiveSizeInBytes           : 38034480
Completed                    : False
CompletionDate               : 1/1/0001 12:00:00 AM
CreationDate                 : 12/13/2018 11:00:14 AM
InventoryRetrievalParameters :
InventorySizeInBytes         : 0
JobDescription               :
JobId                        : op1x...JSbthM
JobOutputPath                :
OutputLocation               :
RetrievalByteRange           : 0-38034479
SelectParameters             :
```

```
SHA256TreeHash              : 79f3ea754c02f58...dc57bf4395b
SNSTopic                    :
StatusCode                  : InProgress
StatusMessage               :
Tier                        : Standard
VaultARN                    : arn:aws:glacier:us-west-2:012345678912:vaults/test
```

- Pour API plus de détails, consultez la section DescribeJobRéférence des AWS Tools for PowerShell applets de commande.

### `New-GLCVault`

L'exemple de code suivant montre comment utiliser`New-GLCVault`.

Outils pour PowerShell

Exemple 1 : crée un nouveau coffre-fort pour le compte de l'utilisateur. Comme aucune valeur n'a été fournie au AccountId paramètre -, les applets de commande utilisent la valeur par défaut « - » pour indiquer le compte courant.

```
New-GLCVault -VaultName myvault
```

Sortie :

```
/01234567812/vaults/myvault
```

- Pour API plus de détails, consultez la section CreateVaultRéférence des AWS Tools for PowerShell applets de commande.

### `Read-GLCJobOutput`

L'exemple de code suivant montre comment utiliser`Read-GLCJobOutput`.

Outils pour PowerShell

Exemple 1 : télécharge le contenu de l'archive dont l'extraction était planifiée dans le cadre de la tâche spécifiée et stocke le contenu dans un fichier sur disque. Le téléchargement valide le checksum pour vous, s'il est disponible. Si nécessaire, la somme de contrôle peut être obtenue à partir de l'historique des réponses du service comme suit (en supposant que cette applet de commande date de la dernière exécution) :. **`$AWSHistory.LastServiceResponse`** Si l'applet

de commande n'a pas été exécutée le plus récemment, inspectez la **`$AWSHistory.Commands`** collection pour obtenir la réponse du service appropriée.

```
Read-GLCJobOutput -VaultName myvault -JobId "HSWjArc...Zq2XLiW" -FilePath "c:\temp
\blue.bin"
```

- Pour API plus de détails, consultez la section GetJobOutputRéférence des AWS Tools for PowerShell applets de commande.

### `Start-GLCJob`

L'exemple de code suivant montre comment utiliser`Start-GLCJob`.

Outils pour PowerShell

Exemple 1 : Démarre une tâche pour récupérer une archive dans le coffre spécifié appartenant à l'utilisateur. L'état de la tâche peut être vérifié à l'aide de l'GLCJobapplet de commande Get-. Lorsque le travail est terminé avec succès, l'GCJobOutputapplet de commande Read- peut être utilisée pour récupérer le contenu de l'archive dans le système de fichiers local.

```
Start-GLCJob -VaultName myvault -JobType "archive-retrieval" -JobDescription
 "archive retrieval" -ArchiveId "o9O9j...TX-TpIhQJw"
```

Sortie :

```
JobId            JobOutputPath Location
-----            ------------- --------
op1x...JSbthM                  /012345678912/vaults/test/jobs/op1xe...I4HqCHkSJSbthM
```

- Pour API plus de détails, consultez la section InitiateJobRéférence des AWS Tools for PowerShell applets de commande.

### `Write-GLCArchive`

L'exemple de code suivant montre comment utiliser`Write-GLCArchive`.

Outils pour PowerShell

Exemple 1 : télécharge un seul fichier dans le coffre spécifié, en renvoyant l'ID de l'archive et la somme de contrôle calculée.

```
Write-GLCArchive -VaultName myvault -FilePath c:\temp\blue.bin
```

Sortie :

```
FilePath                  ArchiveId             Checksum
--------                  ---------             --------
C:\temp\blue.bin          o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
```

Exemple 2 : télécharge le contenu d'une hiérarchie de dossiers dans le coffre spécifié dans le compte de l'utilisateur. Pour chaque fichier chargé, l'applet de commande émet le nom du fichier, l'ID d'archive correspondant et la somme de contrôle calculée de l'archive.

```
Write-GLCArchive -VaultName myvault -FolderPath . -Recurse
```

Sortie :

```
FilePath                  ArchiveId             Checksum
--------                  ---------             --------
C:\temp\blue.bin          o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
C:\temp\green.bin         qXAfOdSG...czo729UHXrw d50a1...9184b9
C:\temp\lum.bin           39aNifP3...q9nb8nZkFIg 28886...5c3e27
C:\temp\red.bin           vp7E6rU_...Ejk_HhjAxKA e05f7...4e34f5
C:\temp\Folder1\file1.txt _eRINlip...5Sxy7dD2BaA d0d2a...c8a3ba
C:\temp\Folder2\file2.iso -Ix3jlmu...iXiDh-XfOPA 7469e...3e86f1
```

- Pour API plus de détails, consultez la section UploadArchiveRéférence des AWS Tools for PowerShell applets de commande.

# SESExemples Amazon utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS Tools for PowerShell aide d'AmazonSES.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

# Actions

### `Get-SESIdentity`

L'exemple de code suivant montre comment utiliser`Get-SESIdentity`.

Outils pour PowerShell

Exemple 1 : Cette commande renvoie une liste contenant toutes les identités (adresses e-mail et domaines) d'un AWS compte spécifique, quel que soit le statut de vérification.

```
Get-SESIdentity
```

- Pour API plus de détails, consultez la section ListIdentitiesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SESSendQuota`

L'exemple de code suivant montre comment utiliser`Get-SESSendQuota`.

Outils pour PowerShell

Exemple 1 : Cette commande renvoie les limites d'envoi actuelles de l'utilisateur.

```
Get-SESSendQuota
```

- Pour API plus de détails, consultez la section GetSendQuotaRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SESSendStatistic`

L'exemple de code suivant montre comment utiliser`Get-SESSendStatistic`.

Outils pour PowerShell

Exemple 1 : Cette commande renvoie les statistiques d'envoi de l'utilisateur. Le résultat est une liste de points de données représentant les deux dernières semaines d'activité d'envoi. Chaque point de données de la liste contient des statistiques pour un intervalle de 15 minutes.

```
Get-SESSendStatistic
```

- Pour API plus de détails, consultez la section GetSendStatisticsRéférence des AWS Tools for PowerShell applets de commande.

# SNSExemples Amazon utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS Tools for PowerShell aide d'AmazonSNS.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `Publish-SNSMessage`

L'exemple de code suivant montre comment utiliser`Publish-SNSMessage`.

Outils pour PowerShell

Exemple 1 : Cet exemple montre la publication d'un message avec un seul élément MessageAttribute déclaré en ligne.

```
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -Message
 "Hello" -MessageAttribute
```

```
 @{'City'=[Amazon.SimpleNotificationService.Model.MessageAttributeValue]@{DataType='String';
 StringValue ='AnyCity'}}
```

Exemple 2 : Cet exemple montre la publication d'un message dont plusieurs MessageAttributes ont été déclarés à l'avance.

```
$cityAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$cityAttributeValue.DataType = "String"
$cityAttributeValue.StringValue = "AnyCity"

$populationAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$populationAttributeValue.DataType = "Number"
$populationAttributeValue.StringValue = "1250800"

$messageAttributes = New-Object System.Collections.Hashtable
$messageAttributes.Add("City", $cityAttributeValue)
$messageAttributes.Add("Population", $populationAttributeValue)

Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -Message
 "Hello" -MessageAttribute $messageAttributes
```

- Pour API plus de détails, voir Publier dans la référence des AWS Tools for PowerShell applets de commande.

# SQSExemples Amazon utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS Tools for PowerShell aide d'AmazonSQS.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

# Actions

### `Add-SQSPermission`

L'exemple de code suivant montre comment utiliser`Add-SQSPermission`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet à la personne spécifiée Compte AWS d'envoyer des messages depuis la file d'attente spécifiée.

```
Add-SQSPermission -Action SendMessage -AWSAccountId 80398EXAMPLE -Label
 SendMessagesFromMyQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue
```

- Pour API plus de détails, consultez la section AddPermissionRéférence des AWS Tools for PowerShell applets de commande.

### `Clear-SQSQueue`

L'exemple de code suivant montre comment utiliser`Clear-SQSQueue`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime tous les messages de la file d'attente spécifiée.

```
Clear-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Pour API plus de détails, consultez la section PurgeQueueRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-SQSMessageVisibility`

L'exemple de code suivant montre comment utiliser`Edit-SQSMessageVisibility`.

Outils pour PowerShell

Exemple 1 : Cet exemple modifie le délai de visibilité du message avec l'identifiant de réception spécifié dans la file d'attente spécifiée à 10 heures (10 heures* 60 minutes* 60 secondes = 36 000 secondes).

```
Edit-SQSMessageVisibility -QueueUrl https://sqs.us-east-1.amazonaws.com/8039EXAMPLE/
MyQueue -ReceiptHandle AQEBgGDh...J/Iqww== -VisibilityTimeout 36000
```

- Pour API plus de détails, consultez la section ChangeMessageVisibilityRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-SQSMessageVisibilityBatch`

L'exemple de code suivant montre comment utiliser`Edit-SQSMessageVisibilityBatch`.

Outils pour PowerShell

Exemple 1 : Cet exemple modifie le délai de visibilité pour 2 messages avec les descripteurs de réception spécifiés dans la file d'attente spécifiée. Le délai de visibilité du premier message passe à 10 heures (10 heures* 60 minutes* 60 secondes = 36 000 secondes). Le délai de visibilité du second message passe à 5 heures (5 heures* 60 minutes* 60 secondes = 18 000 secondes).

```
$changeVisibilityRequest1 = New-Object
 Amazon.SQS.Model.ChangeMessageVisibilityBatchRequestEntry
$changeVisibilityRequest1.Id = "Request1"
$changeVisibilityRequest1.ReceiptHandle = "AQEBd329...v6gl8Q=="
$changeVisibilityRequest1.VisibilityTimeout = 36000

$changeVisibilityRequest2 = New-Object
 Amazon.SQS.Model.ChangeMessageVisibilityBatchRequestEntry
$changeVisibilityRequest2.Id = "Request2"
$changeVisibilityRequest2.ReceiptHandle = "AQEBgGDh...J/Iqww=="
$changeVisibilityRequest2.VisibilityTimeout = 18000

Edit-SQSMessageVisibilityBatch -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue -Entry $changeVisibilityRequest1,
 $changeVisibilityRequest2
```

Sortie :

```
Failed    Successful

------    ----------

{}        {Request2, Request1}
```

- Pour API plus de détails, consultez la section ChangeMessageVisibilityBatchRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SQSDeadLetterSourceQueue`

L'exemple de code suivant montre comment utiliser`Get-SQSDeadLetterSourceQueue`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie toutes les URLs files d'attente qui utilisent la file d'attente spécifiée comme file d'attente de lettres mortes.

```
Get-SQSDeadLetterSourceQueue -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

Sortie :

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
```

- Pour API plus de détails, consultez la section ListDeadLetterSourceQueuesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SQSQueue`

L'exemple de code suivant montre comment utiliser`Get-SQSQueue`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie toutes les files d'attente.

```
Get-SQSQueue
```

Sortie :

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/AnotherQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/DeadLetterQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

Exemple 2 : Cet exemple répertorie toutes les files d'attente qui commencent par le nom spécifié.

```
Get-SQSQueue -QueueNamePrefix My
```

Sortie :

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

- Pour API plus de détails, consultez la section ListQueuesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SQSQueueAttribute`

L'exemple de code suivant montre comment utiliser`Get-SQSQueueAttribute`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie tous les attributs de la file d'attente spécifiée.

```
Get-SQSQueueAttribute -AttributeName All -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Sortie :

```
VisibilityTimeout                     : 30
DelaySeconds                          : 0
MaximumMessageSize                    : 262144
MessageRetentionPeriod                : 345600
ApproximateNumberOfMessages           : 0
ApproximateNumberOfMessagesNotVisible : 0
ApproximateNumberOfMessagesDelayed    : 0
CreatedTimestamp                      : 2/11/2015 5:53:35 PM
LastModifiedTimestamp                 : 12/29/2015 2:23:17 PM
QueueARN                              : arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue
Policy                                :
 {"Version":"2008-10-17","Id":"arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue/
SQSDefaultPolicy","Statement":[{"Sid":"Sid14
                                      495134224EX","Effect":"Allow","Principal":
{"AWS":"*"},"Action":"SQS:SendMessage","Resource":"arn:aws:sqs:us-east-1:80
```

```
                                        398EXAMPLE:MyQueue","Condition":
{"ArnEquals":{"aws:SourceArn":"arn:aws:sns:us-east-1:80398EXAMPLE:MyTopic"}}},
{"Sid":

 "SendMessagesFromMyQueue","Effect":"Allow","Principal":
{"AWS":"80398EXAMPLE"},"Action":"SQS:SendMessage","Resource":"
                                        arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue"}]}
Attributes                             : {[QueueArn, arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue], [ApproximateNumberOfMessages, 0],
                                        [ApproximateNumberOfMessagesNotVisible, 0],
 [ApproximateNumberOfMessagesDelayed, 0]...}
```

Exemple 2 : Cet exemple répertorie séparément uniquement les attributs spécifiés pour la file d'attente spécifiée.

```
Get-SQSQueueAttribute -AttributeName MaximumMessageSize, VisibilityTimeout -QueueUrl
 https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Sortie :

```
VisibilityTimeout                      : 30
DelaySeconds                           : 0
MaximumMessageSize                     : 262144
MessageRetentionPeriod                 : 345600
ApproximateNumberOfMessages            : 0
ApproximateNumberOfMessagesNotVisible : 0
ApproximateNumberOfMessagesDelayed     : 0
CreatedTimestamp                       : 2/11/2015 5:53:35 PM
LastModifiedTimestamp                  : 12/29/2015 2:23:17 PM
QueueARN                               : arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue
Policy                                 :
 {"Version":"2008-10-17","Id":"arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue/
SQSDefaultPolicy","Statement":[{"Sid":"Sid14
                                        495134224EX","Effect":"Allow","Principal":
{"AWS":"*"},"Action":"SQS:SendMessage","Resource":"arn:aws:sqs:us-east-1:80
                                        398EXAMPLE:MyQueue","Condition":
{"ArnEquals":{"aws:SourceArn":"arn:aws:sns:us-east-1:80398EXAMPLE:MyTopic"}}},
{"Sid":

 "SendMessagesFromMyQueue","Effect":"Allow","Principal":
{"AWS":"80398EXAMPLE"},"Action":"SQS:SendMessage","Resource":"
```

```
                                        arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue"}]}
Attributes                             : {[MaximumMessageSize, 262144],
 [VisibilityTimeout, 30]}
```

- Pour API plus de détails, consultez la section GetQueueAttributesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SQSQueueUrl`

L'exemple de code suivant montre comment utiliser`Get-SQSQueueUrl`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie la URL file d'attente portant le nom spécifié.

```
Get-SQSQueueUrl -QueueName MyQueue
```

Sortie :

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Pour API plus de détails, consultez la section GetQueueUrlRéférence des AWS Tools for PowerShell applets de commande.

### `New-SQSQueue`

L'exemple de code suivant montre comment utiliser`New-SQSQueue`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une file d'attente portant le nom spécifié.

```
New-SQSQueue -QueueName MyQueue
```

Sortie :

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Pour API plus de détails, consultez la section CreateQueueRéférence des AWS Tools for PowerShell applets de commande.

### `Receive-SQSMessage`

L'exemple de code suivant montre comment utiliser`Receive-SQSMessage`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie les informations relatives aux 10 prochains messages à recevoir pour la file d'attente spécifiée. Les informations contiendront des valeurs pour les attributs de message spécifiés, s'ils existent.

```
Receive-SQSMessage -AttributeName SenderId, SentTimestamp -MessageAttributeName
 StudentName, StudentGrade -MessageCount 10 -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Sortie :

```
Attributes             : {[SenderId, AIDAIAZKMSNQ7TEXAMPLE], [SentTimestamp,
 1451495923744]}
Body                   : Information about John Doe's grade.
MD5OfBody              : ea572796e3c231f974fe75d89EXAMPLE
MD5OfMessageAttributes : 48c1ee811f0fe7c4e88fbe0f5EXAMPLE
MessageAttributes      : {[StudentGrade, Amazon.SQS.Model.MessageAttributeValue],
 [StudentName, Amazon.SQS.Model.MessageAttributeValue]}
MessageId              : 53828c4b-631b-469b-8833-c093cEXAMPLE
ReceiptHandle          : AQEBpfGp...20Q5cg==
```

- Pour API plus de détails, consultez la section ReceiveMessageRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-SQSMessage`

L'exemple de code suivant montre comment utiliser`Remove-SQSMessage`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime le message avec le descripteur de réception spécifié de la file d'attente spécifiée.

```
Remove-SQSMessage -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
 -ReceiptHandle AQEBd329...v6gl8Q==
```

- Pour API plus de détails, consultez la section DeleteMessageRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-SQSMessageBatch`

L'exemple de code suivant montre comment utiliser`Remove-SQSMessageBatch`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime 2 messages avec les descripteurs de réception spécifiés de la file d'attente spécifiée.

```
$deleteMessageRequest1 = New-Object Amazon.SQS.Model.DeleteMessageBatchRequestEntry
$deleteMessageRequest1.Id = "Request1"
$deleteMessageRequest1.ReceiptHandle = "AQEBX2g4...wtJSQg=="

$deleteMessageRequest2 = New-Object Amazon.SQS.Model.DeleteMessageBatchRequestEntry
$deleteMessageRequest2.Id = "Request2"
$deleteMessageRequest2.ReceiptHandle = "AQEBqOVY...KTsLYg=="

Remove-SQSMessageBatch -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue -Entry $deleteMessageRequest1, $deleteMessageRequest2
```

Sortie :

```
Failed    Successful

------    ----------

{}        {Request1, Request2}
```

- Pour API plus de détails, consultez la section DeleteMessageBatchRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-SQSPermission`

L'exemple de code suivant montre comment utiliser`Remove-SQSPermission`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime les paramètres d'autorisation portant l'étiquette spécifiée de la file d'attente spécifiée.

```
Remove-SQSPermission -Label SendMessagesFromMyQueue -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Pour API plus de détails, consultez la section RemovePermissionRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-SQSQueue`

L'exemple de code suivant montre comment utiliser`Remove-SQSQueue`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime la file d'attente spécifiée.

```
Remove-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Pour API plus de détails, consultez la section DeleteQueueRéférence des AWS Tools for PowerShell applets de commande.

### `Send-SQSMessage`

L'exemple de code suivant montre comment utiliser`Send-SQSMessage`.

Outils pour PowerShell

Exemple 1 : Cet exemple envoie un message avec les attributs et le corps du message spécifiés à la file d'attente spécifiée avec un délai de livraison du message de 10 secondes.

```
$cityAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$cityAttributeValue.DataType = "String"
$cityAttributeValue.StringValue = "AnyCity"

$populationAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$populationAttributeValue.DataType = "Number"
$populationAttributeValue.StringValue = "1250800"
```

```
$messageAttributes = New-Object System.Collections.Hashtable
$messageAttributes.Add("City", $cityAttributeValue)
$messageAttributes.Add("Population", $populationAttributeValue)

Send-SQSMessage -DelayInSeconds 10 -MessageAttributes $messageAttributes -
MessageBody "Information about the largest city in Any Region." -QueueUrl https://
sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Sortie :

```
MD5OfMessageAttributes             MD5OfMessageBody                    MessageId

----------------------             ----------------                    ---------

1d3e51347bc042efbdf6dda31EXAMPLE   51b0a3256d59467f973009b73EXAMPLE    c35fed8f-
c739-4d0c-818b-1820eEXAMPLE
```

- Pour API plus de détails, consultez la section SendMessageRéférence des AWS Tools for PowerShell applets de commande.

### `Send-SQSMessageBatch`

L'exemple de code suivant montre comment utiliser`Send-SQSMessageBatch`.

Outils pour PowerShell

Exemple 1 : Cet exemple envoie 2 messages avec les attributs et les corps de message spécifiés à la file d'attente spécifiée. La livraison est retardée de 15 secondes pour le premier message et de 10 secondes pour le second.

```
$student1NameAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student1NameAttributeValue.DataType = "String"
$student1NameAttributeValue.StringValue = "John Doe"

$student1GradeAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student1GradeAttributeValue.DataType = "Number"
$student1GradeAttributeValue.StringValue = "89"

$student2NameAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student2NameAttributeValue.DataType = "String"
$student2NameAttributeValue.StringValue = "Jane Doe"
```

```
$student2GradeAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student2GradeAttributeValue.DataType = "Number"
$student2GradeAttributeValue.StringValue = "93"

$message1 = New-Object Amazon.SQS.Model.SendMessageBatchRequestEntry
$message1.DelaySeconds = 15
$message1.Id = "FirstMessage"
$message1.MessageAttributes.Add("StudentName", $student1NameAttributeValue)
$message1.MessageAttributes.Add("StudentGrade", $student1GradeAttributeValue)
$message1.MessageBody = "Information about John Doe's grade."

$message2 = New-Object Amazon.SQS.Model.SendMessageBatchRequestEntry
$message2.DelaySeconds = 10
$message2.Id = "SecondMessage"
$message2.MessageAttributes.Add("StudentName", $student2NameAttributeValue)
$message2.MessageAttributes.Add("StudentGrade", $student2GradeAttributeValue)
$message2.MessageBody = "Information about Jane Doe's grade."

Send-SQSMessageBatch -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue -Entry $message1, $message2
```

Sortie :

```
Failed    Successful

------    ----------

{}        {FirstMessage, SecondMessage}
```

- Pour API plus de détails, consultez la section SendMessageBatchRéférence des AWS Tools for PowerShell applets de commande.

### `Set-SQSQueueAttribute`

L'exemple de code suivant montre comment utiliser`Set-SQSQueueAttribute`.

Outils pour PowerShell

Exemple 1 : Cet exemple montre comment définir une politique d'abonnement à une file d'attente à un SNS sujet. Lorsqu'un message est publié dans le sujet, un message est envoyé à la file d'attente abonnée.

```
# create the queue and topic to be associated
$qurl = New-SQSQueue -QueueName "myQueue"
$topicarn = New-SNSTopic -Name "myTopic"

# get the queue ARN to inject into the policy; it will be returned
# in the output's QueueARN member but we need to put it into a variable
# so text expansion in the policy string takes effect
$qarn = (Get-SQSQueueAttribute -QueueUrl $qurl -AttributeName "QueueArn").QueueARN

# construct the policy and inject arns
$policy = @"
{
  "Version": "2008-10-17",
  "Id": "$qarn/SQSPOLICY",
  "Statement": [
      {
      "Sid": "1",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "SQS:SendMessage",
      "Resource": "$qarn",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "$topicarn"
          }
      }
    }
  ]
}
"@

# set the policy
Set-SQSQueueAttribute -QueueUrl $qurl -Attribute @{ Policy=$policy }
```

Exemple 2 : Cet exemple définit les attributs spécifiés pour la file d'attente spécifiée.

```
Set-SQSQueueAttribute -Attribute @{"DelaySeconds" = "10"; "MaximumMessageSize" =
 "131072"} -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Pour API plus de détails, consultez la section SetQueueAttributesRéférence des AWS Tools for PowerShell applets de commande.

# AWS STS exemples utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS Tools for PowerShell with AWS STS.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques



## Actions

### `Convert-STSAuthorizationMessage`

L'exemple de code suivant montre comment utiliser`Convert-STSAuthorizationMessage`.

Outils pour PowerShell

Exemple 1 : Décode les informations supplémentaires contenues dans le contenu du message codé fourni qui a été renvoyé en réponse à une demande. Les informations supplémentaires sont codées car les détails du statut d'autorisation peuvent constituer des informations privilégiées que l'utilisateur qui a demandé l'action ne doit pas voir.

```
Convert-STSAuthorizationMessage -EncodedMessage "...encoded message..."
```

- Pour API plus de détails, consultez la section DecodeAuthorizationMessageRéférence des AWS Tools for PowerShell applets de commande.

### `Get-STSFederationToken`

L'exemple de code suivant montre comment utiliser`Get-STSFederationToken`.

Outils pour PowerShell

Exemple 1 : demande un jeton fédéré valide pendant une heure en utilisant « Bob » comme nom de l'utilisateur fédéré. Ce nom peut être utilisé pour faire référence au nom d'utilisateur fédéré dans une politique basée sur les ressources (telle qu'une politique de compartiment Amazon S3). La IAM politique fournie, sous JSON forme de format, est utilisée pour limiter les autorisations disponibles pour l'IAMutilisateur. La politique fournie ne peut pas accorder plus d'autorisations que celles accordées à l'utilisateur demandeur, les autorisations finales pour l'utilisateur fédéré étant l'ensemble le plus restrictif en fonction de l'intersection entre la politique adoptée et la politique IAM utilisateur.

```
Get-STSFederationToken -Name "Bob" -Policy "...JSON policy..." -DurationInSeconds
 3600
```

- Pour API plus de détails, consultez la section GetFederationTokenRéférence des AWS Tools for PowerShell applets de commande.

### `Get-STSSessionToken`

L'exemple de code suivant montre comment utiliser`Get-STSSessionToken`.

Outils pour PowerShell

Exemple 1 : renvoie une **`Amazon.RuntimeAWSCredentials`** instance contenant des informations d'identification temporaires valides pendant une période définie. Les informations d'identification utilisées pour demander des informations d'identification temporaires sont déduites des paramètres par défaut actuels du shell. Pour spécifier d'autres informations d'identification, utilisez les SecretKey paramètres - ProfileName ou - AccessKey /-.

```
Get-STSSessionToken
```

Sortie :

```
AccessKeyId                          Expiration
 SecretAccessKey                     SessionToken
-----------                          ----------
 ---------------                     ------------
EXAMPLEACCESSKEYID                   2/16/2015 9:12:28 PM
 examplesecretaccesskey...           SamPleTokeN.....
```

Exemple 2 : renvoie une **`Amazon.RuntimeAWSCredentials`** instance contenant des informations d'identification temporaires valides pendant une heure. Les informations d'identification utilisées pour effectuer la demande sont obtenues à partir du profil spécifié.

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile
```

Sortie :

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

Exemple 3 : renvoie une **`Amazon.RuntimeAWSCredentials`** instance contenant des informations d'identification temporaires valides pendant une heure en utilisant le numéro d'identification de l'MFAappareil associé au compte dont les informations d'identification sont spécifiées dans le profil « myprofilename » et la valeur fournie par l'appareil.

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile -SerialNumber
 YourMFADeviceSerialNumber -TokenCode 123456
```

Sortie :

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

- Pour API plus de détails, consultez la section GetSessionTokenRéférence des AWS Tools for PowerShell applets de commande.

### `Use-STSRole`

L'exemple de code suivant montre comment utiliser`Use-STSRole`.

Exemple 1 : renvoie un ensemble d'informations d'identification temporaires (clé d'accès, clé secrète et jeton de session) qui peuvent être utilisées pendant une heure pour accéder à AWS des ressources auxquelles l'utilisateur demandeur n'a pas normalement accès. Les informations d'identification renvoyées disposent des autorisations autorisées par la politique d'accès du rôle assumé et par la politique fournie (vous ne pouvez pas utiliser la politique fournie pour accorder des autorisations supérieures à celles définies par la politique d'accès du rôle assumé).

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
Policy "...JSON policy..." -DurationInSeconds 3600
```

Exemple 2 : renvoie un ensemble d'informations d'identification temporaires, valides pendant une heure, dotées des mêmes autorisations que celles définies dans la politique d'accès du rôle assumé.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
DurationInSeconds 3600
```

Exemple 3 : renvoie un ensemble d'informations d'identification temporaires fournissant le numéro de série et le jeton généré à partir d'un identifiant MFA associé aux informations d'identification utilisateur utilisées pour exécuter l'applet de commande.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
DurationInSeconds 3600 -SerialNumber "GAHT12345678" -TokenCode "123456"
```

Exemple 4 : renvoie un ensemble d'informations d'identification temporaires qui ont assumé un rôle défini dans un compte client. Pour chaque rôle que le tiers peut assumer, le compte client doit créer un rôle à l'aide d'un identifiant qui doit être transmis dans le ExternalId paramètre - chaque fois que le rôle est assumé.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
DurationInSeconds 3600 -ExternalId "ABC123"
```

- Pour API plus de détails, consultez la section AssumeRoleRéférence des AWS Tools for PowerShell applets de commande.

### `Use-STSWebIdentityRole`

L'exemple de code suivant montre comment utiliser`Use-STSWebIdentityRole`.

Outils pour PowerShell

Exemple 1 : renvoie un ensemble temporaire d'informations d'identification, valides pendant une heure, pour un utilisateur authentifié auprès du fournisseur d'identité Login with Amazon. Les informations d'identification reposent sur la politique d'accès associée au rôle identifié par le rôleARN. Vous pouvez éventuellement transmettre une JSON politique au paramètre -Policy pour affiner davantage les autorisations d'accès (vous ne pouvez pas accorder plus d'autorisations que celles disponibles dans les autorisations associées au rôle). La valeur fournie à - WebIdentityToken est l'identifiant utilisateur unique renvoyé par le fournisseur d'identité.

```
Use-STSWebIdentityRole -DurationInSeconds 3600 -ProviderId "www.amazon.com"
 -RoleSessionName "app1" -RoleArn "arn:aws:iam::123456789012:role/
FederatedWebIdentityRole" -WebIdentityToken "Atza...DVI0r1"
```

- Pour API plus de détails, consultez la section AssumeRoleWithWebIdentityRéférence des AWS Tools for PowerShell applets de commande.

## AWS Support exemples utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS Tools for PowerShell with AWS Support.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

# Actions

### `Add-ASACommunicationToCase`

L'exemple de code suivant montre comment utiliser`Add-ASACommunicationToCase`.

Outils pour PowerShell

Exemple 1 : ajoute le corps d'une communication par e-mail au cas spécifié.

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -
CommunicationBody "Some text about the case"
```

Exemple 2 : ajoute le corps d'une communication par e-mail au cas spécifié, plus une ou plusieurs adresses e-mail contenues dans la ligne CC de l'e-mail.

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -
CcEmailAddress @("email1@address.com", "email2@address.com") -CommunicationBody
 "Some text about the case"
```

- Pour API plus de détails, consultez la section AddCommunicationToCaseRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ASACase`

L'exemple de code suivant montre comment utiliser`Get-ASACase`.

Outils pour PowerShell

Exemple 1 : renvoie les détails de tous les dossiers de support.

```
Get-ASACase
```

Exemple 2 : renvoie les détails de tous les dossiers de support depuis la date et l'heure spécifiées.

```
Get-ASACase -AfterTime "2013-09-10T03:06Z"
```

Exemple 3 : renvoie les détails des 10 premiers cas de support, y compris ceux qui ont été résolus.

```
Get-ASACase -MaxResult 10 -IncludeResolvedCases $true
```

Exemple 4 : renvoie les détails de l'unique dossier de support spécifié.

```
Get-ASACase -CaseIdList "case-12345678910-2013-c4c1d2bf33c5cf47"
```

Exemple 5 : renvoie les détails des demandes de support spécifiées.

```
Get-ASACase -CaseIdList @("case-12345678910-2013-c4c1d2bf33c5cf47",
 "case-18929034710-2011-c4fdeabf33c5cf47")
```

Exemple 6 : renvoie tous les dossiers de support à l'aide de la pagination manuelle. Les étuis sont récupérés par lots de 20.

```
$nextToken = $null
do {
  Get-ASACase -NextToken $nextToken -MaxResult 20
  $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Pour API plus de détails, consultez la section DescribeCasesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ASACommunication`

L'exemple de code suivant montre comment utiliser`Get-ASACommunication`.

Outils pour PowerShell

Exemple 1 : renvoie toutes les communications pour le cas spécifié.

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

Exemple 2 : renvoie toutes les communications depuis minuit UTC le 1er janvier 2012 pour le cas spécifié.

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -AfterTime
 "2012-01-10T00:00Z"
```

Exemple 3 : renvoie toutes les communications depuis minuit UTC le 1er janvier 2012 pour le cas spécifié, en utilisant la pagination manuelle. Les communications sont récupérées par lots de 20.

```
$nextToken = $null
do {
  Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -NextToken
 $nextToken -MaxResult 20
  $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Pour API plus de détails, consultez la section DescribeCommunicationsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ASAService`

L'exemple de code suivant montre comment utiliser`Get-ASAService`.

Outils pour PowerShell

Exemple 1 : renvoie tous les codes de service, noms et catégories disponibles.

```
Get-ASAService
```

Exemple 2 : renvoie le nom et les catégories du service avec le code spécifié.

```
Get-ASAService -ServiceCodeList "amazon-cloudfront"
```

Exemple 3 : renvoie le nom et les catégories des codes de service spécifiés.

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch")
```

Exemple 4 : renvoie le nom et les catégories (en japonais) pour les codes de service spécifiés. Les codes de langue anglais (« en ») et japonais (« ja ») sont actuellement pris en charge.

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch") -
Language "ja"
```

- Pour API plus de détails, consultez la section DescribeServicesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ASASeverityLevel`

L'exemple de code suivant montre comment utiliser`Get-ASASeverityLevel`.

Outils pour PowerShell

Exemple 1 : renvoie la liste des niveaux de gravité qui peuvent être attribués à un dossier AWS Support.

```
Get-ASASeverityLevel
```

Exemple 2 : renvoie la liste des niveaux de gravité qui peuvent être attribués à un dossier AWS Support. Les noms des niveaux sont renvoyés en japonais.

```
Get-ASASeverityLevel -Language "ja"
```

- Pour API plus de détails, consultez la section DescribeSeverityLevelsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ASATrustedAdvisorCheck`

L'exemple de code suivant montre comment utiliser`Get-ASATrustedAdvisorCheck`.

Outils pour PowerShell

Exemple 1 : Renvoie la collection de chèques Trusted Advisor. Vous devez spécifier le paramètre Language qui peut accepter soit « en » pour la sortie en anglais, soit « ja » pour la sortie japonaise.

```
Get-ASATrustedAdvisorCheck -Language "en"
```

- Pour API plus de détails, consultez la section DescribeTrustedAdvisorChecksRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ASATrustedAdvisorCheckRefreshStatus`

L'exemple de code suivant montre comment utiliser`Get-ASATrustedAdvisorCheckRefreshStatus`.

Outils pour PowerShell

Exemple 1 : renvoie l'état actuel des demandes d'actualisation pour les vérifications spécifiées. Request- ASATrustedAdvisorCheckRefresh peut être utilisé pour demander que les informations d'état des chèques soient actualisées.

```
Get-ASATrustedAdvisorCheckRefreshStatus -CheckId @("checkid1", "checkid2")
```

- Pour API plus de détails, consultez la section DescribeTrustedAdvisorCheckRefreshStatusesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ASATrustedAdvisorCheckResult`

L'exemple de code suivant montre comment utiliser`Get-ASATrustedAdvisorCheckResult`.

Outils pour PowerShell

Exemple 1 : renvoie les résultats d'un check Trusted Advisor. La liste des chèques Trusted Advisor disponibles peut être obtenue à l'aide de Get-ASATrustedAdvisorChecks. Le résultat est l'état général de la vérification, l'horodatage auquel la vérification a été exécutée pour la dernière fois et l'identifiant de contrôle unique pour la vérification spécifique. Pour que les résultats soient affichés en japonais, ajoutez le paramètre -Language « ja ».

```
Get-ASATrustedAdvisorCheckResult -CheckId "checkid1"
```

- Pour API plus de détails, consultez la section DescribeTrustedAdvisorCheckResultRéférence des AWS Tools for PowerShell applets de commande.

### `Get-ASATrustedAdvisorCheckSummary`

L'exemple de code suivant montre comment utiliser`Get-ASATrustedAdvisorCheckSummary`.

Outils pour PowerShell

Exemple 1 : Renvoie le dernier résumé du check Trusted Advisor spécifié.

```
Get-ASATrustedAdvisorCheckSummary -CheckId "checkid1"
```

Exemple 2 : renvoie les derniers résumés des contrôles Trusted Advisor spécifiés.

```
Get-ASATrustedAdvisorCheckSummary -CheckId @("checkid1", "checkid2")
```

- Pour API plus de détails, consultez la section DescribeTrustedAdvisorCheckSummariesRéférence des AWS Tools for PowerShell applets de commande.

### `New-ASACase`

L'exemple de code suivant montre comment utiliser`New-ASACase`.

Outils pour PowerShell

Exemple 1 : Crée un nouveau dossier dans le AWS Support Center. Les valeurs des CategoryCode paramètres - ServiceCode et - peuvent être obtenues à l'aide de l'ASAServiceapplet de commande Get-. La valeur du SeverityCode paramètre - peut être obtenue à l'aide de l'ASASeverityLevelapplet de commande Get-. La valeur du IssueType paramètre - peut être « service client » ou « technique ». En cas de succès, le numéro de dossier de AWS support est affiché. Par défaut, le dossier sera traité en anglais. Pour utiliser le japonais, ajoutez le paramètre -Language « ja ». Les CommunicationBody paramètres -ServiceCode, -CategoryCode, -Subject et - sont obligatoires.

```
New-ASACase -ServiceCode "amazon-cloudfront" -CategoryCode "APIs" -SeverityCode
 "low" -Subject "subject text" -CommunicationBody "description of the case" -
CcEmailAddress @("email1@domain.com", "email2@domain.com") -IssueType "technical"
```

- Pour API plus de détails, consultez la section CreateCaseRéférence des AWS Tools for PowerShell applets de commande.

### `Request-ASATrustedAdvisorCheckRefresh`

L'exemple de code suivant montre comment utiliser`Request-ASATrustedAdvisorCheckRefresh`.

Outils pour PowerShell

Exemple 1 : Demande une actualisation pour le check Trusted Advisor spécifié.

```
Request-ASATrustedAdvisorCheckRefresh -CheckId "checkid1"
```

- Pour API plus de détails, consultez la section RefreshTrustedAdvisorCheckRéférence des AWS Tools for PowerShell applets de commande.

#### `Resolve-ASACase`

L'exemple de code suivant montre comment utiliser`Resolve-ASACase`.

Outils pour PowerShell

Exemple 1 : renvoie l'état initial du cas spécifié et l'état actuel une fois l'appel pour le résoudre terminé.

```
Resolve-ASACase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

- Pour API plus de détails, consultez la section ResolveCaseRéférence des AWS Tools for PowerShell applets de commande.

## Exemples de Systems Manager utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide de AWS Tools for PowerShell with Systems Manager.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

### Actions

#### `Add-SSMResourceTag`

L'exemple de code suivant montre comment utiliser`Add-SSMResourceTag`.

Outils pour PowerShell

Exemple 1 : Cet exemple met à jour une fenêtre de maintenance avec de nouvelles balises. Il n'y a pas de sortie si la commande réussit. La syntaxe utilisée dans cet exemple nécessite PowerShell la version 3 ou ultérieure.

```
$option1 = @{Key="Stack";Value=@("Production")}
Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -Tag $option1
```

Exemple 2 : Avec PowerShell la version 2, vous devez utiliser New-Object pour créer chaque balise. Il n'y a aucune sortie si la commande aboutit.

```
$tag1 = New-Object Amazon.SimpleSystemsManagement.Model.Tag
$tag1.Key = "Stack"
$tag1.Value = "Production"

Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -Tag $tag1
```

- Pour API plus de détails, consultez la section AddTagsToResourceRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-SSMDocumentPermission`

L'exemple de code suivant montre comment utiliser`Edit-SSMDocumentPermission`.

Outils pour PowerShell

Exemple 1 : Cet exemple ajoute des autorisations de « partage » à tous les comptes associés à un document. Il n'y a aucune sortie si la commande aboutit.

```
Edit-SSMDocumentPermission -Name "RunShellScript" -PermissionType "Share" -
AccountIdsToAdd all
```

Exemple 2 : Cet exemple ajoute des autorisations de « partage » à un compte spécifique pour un document. Il n'y a aucune sortie si la commande aboutit.

```
Edit-SSMDocumentPermission -Name "RunShellScriptNew" -PermissionType "Share" -
AccountIdsToAdd "123456789012"
```

- Pour API plus de détails, consultez la section ModifyDocumentPermissionRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMActivation`

L'exemple de code suivant montre comment utiliser`Get-SSMActivation`.

Outils pour PowerShell

Exemple 1 : Cet exemple fournit des détails sur les activations de votre compte.

```
Get-SSMActivation
```

Sortie :

```
ActivationId        : 08e51e79-1e36-446c-8e63-9458569c1363
CreatedDate         : 3/1/2017 12:01:51 AM
DefaultInstanceName : MyWebServers
Description         :
ExpirationDate      : 3/2/2017 12:01:51 AM
Expired             : False
IamRole             : AutomationRole
RegistrationLimit   : 10
RegistrationsCount  : 0
```

- Pour API plus de détails, consultez la section DescribeActivationsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMAssociation`

L'exemple de code suivant montre comment utiliser`Get-SSMAssociation`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit l'association entre une instance et un document.

```
Get-SSMAssociation -InstanceId "i-0000293ffd8c57862" -Name "AWS-UpdateSSMAgent"
```

Sortie :

```
Name                  : AWS-UpdateSSMAgent
InstanceId            : i-0000293ffd8c57862
Date                  : 2/23/2017 6:55:22 PM
Status.Name           : Pending
Status.Date           : 2/20/2015 8:31:11 AM
Status.Message        : temp_status_change
Status.AdditionalInfo : Additional-Config-Needed
```

- Pour API plus de détails, consultez la section DescribeAssociationRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMAssociationExecution`

L'exemple de code suivant montre comment utiliser`Get-SSMAssociationExecution`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie les exécutions pour l'ID d'association fourni

```
Get-SSMAssociationExecution -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

Sortie :

```
AssociationId         : 123a45a0-c678-9012-3456-78901234db5e
AssociationVersion    : 2
CreatedTime           : 3/2/2019 8:53:29 AM
DetailedStatus        :
ExecutionId           : 123a45a0-c678-9012-3456-78901234db5e
LastExecutionDate     : 1/1/0001 12:00:00 AM
ResourceCountByStatus : {Success=4}
Status                : Success
```

- Pour API plus de détails, consultez la section DescribeAssociationExecutionsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMAssociationExecutionTarget`

L'exemple de code suivant montre comment utiliser`Get-SSMAssociationExecutionTarget`.

Exemple 1 : Cet exemple affiche l'ID de ressource et son statut d'exécution qui font partie des cibles d'exécution de l'association

```
Get-SSMAssociationExecutionTarget -AssociationId 123a45a0-
c678-9012-3456-78901234db5e -ExecutionId 123a45a0-c678-9012-3456-78901234db5e |
 Select-Object ResourceId, Status
```

Sortie :

```
ResourceId           Status
----------           ------
i-0b1b2a3456f7a890b  Success
i-01c12a45d6fc7a89f  Success
i-0a1caf234f56d7dc8  Success
i-012a3fd45af6dbcfe  Failed
i-0ddc1df23c4a5fb67  Success
```

Exemple 2 : Cette commande vérifie l'exécution particulière d'une automatisation particulière depuis hier, où un document de commande est associé. Il vérifie en outre si l'exécution de l'association a échoué, et si c'est le cas, il affichera les détails de l'appel de commande pour l'exécution ainsi que l'identifiant de l'instance

```
$AssociationExecution= Get-SSMAssociationExecutionTarget -
AssociationId 1c234567-890f-1aca-a234-5a678d901cb0 -ExecutionId
 12345ca12-3456-2345-2b45-23456789012 |
    Where-Object {$_.LastExecutionDate -gt (Get-Date -Hour 00 -Minute
 00).AddDays(-1)}

foreach ($execution in $AssociationExecution) {
    if($execution.Status -ne 'Success'){
        Write-Output "There was an issue executing the association
 $($execution.AssociationId) on $($execution.ResourceId)"
        Get-SSMCommandInvocation -CommandId $execution.OutputSource.OutputSourceId -
Detail:$true | Select-Object -ExpandProperty CommandPlugins
    }
}
```

Sortie :

```
There was an issue executing the association 1c234567-890f-1aca-a234-5a678d901cb0 on
 i-0a1caf234f56d7dc8


Name                   : aws:runPowerShellScript
Output                 :
                         ----------ERROR-------
                         failed to run commands: exit status 1
OutputS3BucketName     :
OutputS3KeyPrefix      :
OutputS3Region         : eu-west-1
ResponseCode           : 1
ResponseFinishDateTime : 5/29/2019 11:04:49 AM
ResponseStartDateTime  : 5/29/2019 11:04:49 AM
StandardErrorUrl       :
StandardOutputUrl      :
Status                 : Failed
StatusDetails          : Failed
```

- Pour API plus de détails, consultez la section DescribeAssociationExecutionTargetsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMAssociationList`

L'exemple de code suivant montre comment utiliser`Get-SSMAssociationList`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie toutes les associations associées à une instance. La syntaxe utilisée dans cet exemple nécessite PowerShell la version 3 ou ultérieure.

```
$filter1 = @{Key="InstanceId";Value=@("i-0000293ffd8c57862")}
Get-SSMAssociationList -AssociationFilterList $filter1
```

Sortie :

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion    :
InstanceId         : i-0000293ffd8c57862
LastExecutionDate  : 2/20/2015 8:31:11 AM
Name               : AWS-UpdateSSMAgent
```

```
Overview           : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets            : {InstanceIds}
```

Exemple 2 : Cet exemple répertorie toutes les associations associées à un document de configuration. La syntaxe utilisée dans cet exemple nécessite PowerShell la version 3 ou ultérieure.

```
$filter2 = @{Key="Name";Value=@("AWS-UpdateSSMAgent")}
Get-SSMAssociationList -AssociationFilterList $filter2
```

Sortie :

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion    :
InstanceId         : i-0000293ffd8c57862
LastExecutionDate  : 2/20/2015 8:31:11 AM
Name               : AWS-UpdateSSMAgent
Overview           : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets            : {InstanceIds}
```

Exemple 3 : Avec PowerShell la version 2, vous devez utiliser New-Object pour créer chaque filtre.

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.AssociationFilter
$filter1.Key = "InstanceId"
$filter1.Value = "i-0000293ffd8c57862"

Get-SSMAssociationList -AssociationFilterList $filter1
```

Sortie :

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion    :
InstanceId         : i-0000293ffd8c57862
LastExecutionDate  : 2/20/2015 8:31:11 AM
Name               : AWS-UpdateSSMAgent
Overview           : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets            : {InstanceIds}
```

- Pour API plus de détails, consultez la section ListAssociationsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMAssociationVersionList`

L'exemple de code suivant montre comment utiliser`Get-SSMAssociationVersionList`.

Outils pour PowerShell

Exemple 1 : Cet exemple récupère toutes les versions de l'association fournie.

```
Get-SSMAssociationVersionList -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

Sortie :

```
AssociationId      : 123a45a0-c678-9012-3456-78901234db5e
AssociationName    :
AssociationVersion : 2
ComplianceSeverity :
CreatedDate        : 3/12/2019 9:21:01 AM
DocumentVersion    :
MaxConcurrency     :
MaxErrors          :
Name               : AWS-GatherSoftwareInventory
OutputLocation     :
Parameters         : {}
ScheduleExpression :
Targets            : {InstanceIds}

AssociationId      : 123a45a0-c678-9012-3456-78901234db5e
AssociationName    : test-case-1234567890
AssociationVersion : 1
ComplianceSeverity :
CreatedDate        : 3/2/2019 8:53:29 AM
DocumentVersion    :
MaxConcurrency     :
MaxErrors          :
Name               : AWS-GatherSoftwareInventory
OutputLocation     :
Parameters         : {}
ScheduleExpression : rate(30minutes)
Targets            : {InstanceIds}
```

- Pour API plus de détails, consultez la section ListAssociationVersionsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMAutomationExecution`

L'exemple de code suivant montre comment utiliser`Get-SSMAutomationExecution`.

Outils pour PowerShell

Exemple 1 : Cet exemple affiche les détails d'une exécution d'automatisation.

```
Get-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"
```

Sortie :

```
AutomationExecutionId     : 4105a4fc-f944-11e6-9d32-8fb2db27a909
AutomationExecutionStatus : Failed
DocumentName              : AWS-UpdateLinuxAmi
DocumentVersion           : 1
ExecutionEndTime          : 2/22/2017 9:17:08 PM
ExecutionStartTime        : 2/22/2017 9:17:02 PM
FailureMessage            : Step launchInstance failed maximum allowed times. You
 are not authorized to perform this operation. Encoded
                            authorization failure message:
 B_V2QyyN7NhSZQYpmVzpEc4oSnj2GLTNYnXUHsTbqJkNMoDgubmbtthLmZyaiUYekORIrA42-
fv1x-04q5Fjff6glh
                            Yb6TI5b0GQeeNrpwNvpDzmO-
PSR1swlAbg9fdM9BcNjyrznspUkWpuKu9EC1Ou6v3OXU1KC9nZ7mPlWMFZNkSioQqpwWEvMw-
GZktsQzm67qOhUhBNOLWYhbS
                            pkfiqzY-5nw3S0obx30fhd3EJa5O_-
GjV_a0nFXQJa70ik40bFOrEh3MtCSbrQT6--DvFy_FQ8TKvkIXadyVskeJI84XOF5WmA60f1pi5GI08i-
nRfZS6oDeU
                            gELBjjoFKD8s3L2aI0B6umWVxnQOjqhQRxwJ53b54sZJ2PW3v_mtg9-
q0CK0ezS3xfh_y0ilaUGOAZG-xjQFuvU_JZedWpla3xi-MZsmblAifBI
                            (Service: AmazonEC2; Status Code: 403; Error Code:
 UnauthorizedOperation; Request ID:
                            6a002f94-ba37-43fd-99e6-39517715fce5)
Outputs                   : {[createImage.ImageId,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
Parameters                : {[AutomationAssumeRole,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]], [InstanceIamRole,
```

```
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]], [SourceAmiId,

 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
StepExecutions            : {launchInstance, updateOSSoftware, stopInstance,
 createImage...}
```

Exemple 2 : Cet exemple répertorie les détails des étapes pour l'identifiant d'exécution automatique donné

```
Get-SSMAutomationExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object -ExpandProperty StepExecutions | Select-Object
 StepName, Action, StepStatus, ValidNextSteps
```

Sortie :

```
StepName                Action                 StepStatus ValidNextSteps
--------                ------                 ---------- --------------
LaunchInstance          aws:runInstances       Success    {OSCompatibilityCheck}
OSCompatibilityCheck    aws:runCommand         Success    {RunPreUpdateScript}
RunPreUpdateScript      aws:runCommand         Success    {UpdateEC2Config}
UpdateEC2Config         aws:runCommand         Cancelled  {}
UpdateSSMAgent          aws:runCommand         Pending    {}
UpdateAWSPVDriver       aws:runCommand         Pending    {}
UpdateAWSEnaNetworkDriver aws:runCommand       Pending    {}
UpdateAWSNVMe           aws:runCommand         Pending    {}
InstallWindowsUpdates   aws:runCommand         Pending    {}
RunPostUpdateScript     aws:runCommand         Pending    {}
RunSysprepGeneralize    aws:runCommand         Pending    {}
StopInstance            aws:changeInstanceState Pending   {}
CreateImage             aws:createImage        Pending    {}
TerminateInstance       aws:changeInstanceState Pending   {}
```

- Pour API plus de détails, consultez la section GetAutomationExecutionRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMAutomationExecutionList`

L'exemple de code suivant montre comment utiliser`Get-SSMAutomationExecutionList`.

Exemple 1 : Cet exemple décrit toutes les exécutions automatisées actives et terminées associées à votre compte.

```
Get-SSMAutomationExecutionList
```

Sortie :

```
AutomationExecutionId     : 4105a4fc-f944-11e6-9d32-8fb2db27a909
AutomationExecutionStatus : Failed
DocumentName              : AWS-UpdateLinuxAmi
DocumentVersion           : 1
ExecutedBy                : admin
ExecutionEndTime          : 2/22/2017 9:17:08 PM
ExecutionStartTime        : 2/22/2017 9:17:02 PM
LogFile                   :
Outputs                   : {[createImage.ImageId,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
```

Exemple 2 : Cet exemple affiche l'ExecutionID, le document, l'horodatage de début/fin de l'exécution pour les exécutions dont la valeur n'est pas « Success » AutomationExecutionStatus

```
Get-SSMAutomationExecutionList | Where-Object AutomationExecutionStatus
 -ne "Success" | Select-Object AutomationExecutionId, DocumentName,
 AutomationExecutionStatus, ExecutionStartTime, ExecutionEndTime | Format-Table -
AutoSize
```

Sortie :

```
AutomationExecutionId                DocumentName
 AutomationExecutionStatus ExecutionStartTime   ExecutionEndTime
--------------------                ------------
 ------------------------ ------------------   ----------------
e1d2bad3-4567-8901-ae23-456c7c8901be AWS-UpdateWindowsAmi
 Cancelled                 4/16/2019 5:37:04 AM 4/16/2019 5:47:29 AM
61234567-a7f8-90e1-2b34-567b8bf9012c Fixed-UpdateAmi
 Cancelled                 4/16/2019 5:33:04 AM 4/16/2019 5:40:15 AM
91234d56-7e89-0ac1-2aee-34ea5d6a7c89 AWS-UpdateWindowsAmi                    Failed
                   4/16/2019 5:22:46 AM 4/16/2019 5:27:29 AM
```

- Pour API plus de détails, consultez la section DescribeAutomationExecutionsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMAutomationStepExecution`

L'exemple de code suivant montre comment utiliser`Get-SSMAutomationStepExecution`.

Outils pour PowerShell

Exemple 1 : Cet exemple affiche des informations sur toutes les exécutions d'étapes actives et terminées dans un flux de travail d'automatisation.

```
Get-SSMAutomationStepExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object StepName, Action, StepStatus
```

Sortie :

```
StepName                 Action                 StepStatus
--------                 ------                 ----------
LaunchInstance           aws:runInstances       Success
OSCompatibilityCheck     aws:runCommand         Success
RunPreUpdateScript       aws:runCommand         Success
UpdateEC2Config          aws:runCommand         Cancelled
UpdateSSMAgent           aws:runCommand         Pending
UpdateAWSPVDriver        aws:runCommand         Pending
UpdateAWSEnaNetworkDriver aws:runCommand        Pending
UpdateAWSNVMe            aws:runCommand         Pending
InstallWindowsUpdates    aws:runCommand         Pending
RunPostUpdateScript      aws:runCommand         Pending
RunSysprepGeneralize     aws:runCommand         Pending
StopInstance             aws:changeInstanceState Pending
CreateImage              aws:createImage        Pending
TerminateInstance        aws:changeInstanceState Pending
```

- Pour API plus de détails, consultez la section DescribeAutomationStepExecutionsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMAvailablePatch`

L'exemple de code suivant montre comment utiliser`Get-SSMAvailablePatch`.

Exemple 1 : Cet exemple permet d'obtenir tous les correctifs disponibles pour Windows Server 2012 dont le niveau de MSRC gravité est Critique. La syntaxe utilisée dans cet exemple nécessite PowerShell la version 3 ou ultérieure.

```
$filter1 = @{Key="PRODUCT";Values=@("WindowsServer2012")}
$filter2 = @{Key="MSRC_SEVERITY";Values=@("Critical")}

Get-SSMAvailablePatch -Filter $filter1,$filter2
```

Sortie :

```
Classification : SecurityUpdates
ContentUrl     : https://support.microsoft.com/en-us/kb/2727528
Description    : A security issue has been identified that could allow an
 unauthenticated remote attacker to compromise your system and gain control
                  over it. You can help protect your system by installing this update
 from Microsoft. After you install this update, you may have to
                  restart your system.
Id             : 1eb507be-2040-4eeb-803d-abc55700b715
KbNumber       : KB2727528
Language       : All
MsrcNumber     : MS12-072
MsrcSeverity   : Critical
Product        : WindowsServer2012
ProductFamily  : Windows
ReleaseDate    : 11/13/2012 6:00:00 PM
Title          : Security Update for Windows Server 2012 (KB2727528)
Vendor         : Microsoft
...
```

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

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "PRODUCT"
$filter1.Values = "WindowsServer2012"
$filter2 = New-Object Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter2.Key = "MSRC_SEVERITY"
$filter2.Values = "Critical"
```

```
Get-SSMAvailablePatch -Filter $filter1,$filter2
```

Exemple 3 : Cet exemple récupère toutes les mises à jour publiées au cours des 20 derniers jours et applicables aux produits correspondant WindowsServer à 2019

```
Get-SSMAvailablePatch | Where-Object ReleaseDate -ge (Get-Date).AddDays(-20) |
 Where-Object Product -eq "WindowsServer2019" | Select-Object ReleaseDate, Product,
 Title
```

Sortie :

```
ReleaseDate         Product           Title
-----------         -------           -----
4/9/2019 5:00:12 PM WindowsServer2019 2019-04 Security Update for Adobe Flash Player
 for Windows Server 2019 for x64-based Systems (KB4493478)
4/9/2019 5:00:06 PM WindowsServer2019 2019-04 Cumulative Update for Windows Server
 2019 for x64-based Systems (KB4493509)
4/2/2019 5:00:06 PM WindowsServer2019 2019-03 Servicing Stack Update for Windows
 Server 2019 for x64-based Systems (KB4493510)
```

- Pour API plus de détails, consultez la section DescribeAvailablePatchesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMCommand`

L'exemple de code suivant montre comment utiliser`Get-SSMCommand`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie toutes les commandes demandées.

```
Get-SSMCommand
```

Sortie :

```
CommandId          : 4b75a163-d39a-4d97-87c9-98ae52c6be35
Comment            : Apply association with id at update time: 4cc73e42-
d5ae-4879-84f8-57e09c0efcd0
```

```
CompletedCount     : 1
DocumentName       : AWS-RefreshAssociation
ErrorCount         : 0
ExpiresAfter       : 2/24/2017 3:19:08 AM
InstanceIds        : {i-0cb2b964d3e14fd9f}
MaxConcurrency     : 50
MaxErrors          : 0
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
OutputS3BucketName :
OutputS3KeyPrefix  :
OutputS3Region     :
Parameters         : {[associationIds,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
RequestedDateTime  : 2/24/2017 3:18:08 AM
ServiceRole        :
Status             : Success
StatusDetails      : Success
TargetCount        : 1
Targets            : {}
```

Exemple 2 : Cet exemple permet d'obtenir le statut d'une commande spécifique.

```
Get-SSMCommand -CommandId "4b75a163-d39a-4d97-87c9-98ae52c6be35"
```

Exemple 3 : Cet exemple récupère toutes les SSM commandes invoquées après 2019-04-01T 00:00:00 Z

```
Get-SSMCommand -Filter @{Key="InvokedAfter";Value="2019-04-01T00:00:00Z"} | Select-
Object CommandId, DocumentName, Status, RequestedDateTime | Sort-Object -Property
 RequestedDateTime -Descending
```

Sortie :

```
CommandId                            DocumentName               Status
 RequestedDateTime
---------                            ------------               ------
 -----------------
edb1b23e-456a-7adb-aef8-90e-012ac34f AWS-RunPowerShellScript    Cancelled 4/16/2019
 5:45:23 AM
1a2dc3fb-4567-890d-a1ad-234b5d6bc7d9 AWS-ConfigureAWSPackage    Success   4/6/2019
 9:19:42 AM
```

```
12c3456c-7e90-4f12-1232-1234f5b67893 KT-Retrieve-Cloud-Type-Win Failed    4/2/2019
 4:13:07 AM
fe123b45-240c-4123-a2b3-234bdd567ecf AWS-RunInspecChecks        Failed    4/1/2019
 2:27:31 PM
1eb23aa4-567d-4123-12a3-4c1c2ab34561 AWS-RunPowerShellScript    Success   4/1/2019
 1:05:55 PM
1c2f3bb4-ee12-4bc1-1a23-12345eea123e AWS-RunInspecChecks        Failed    4/1/2019
 11:13:09 AM
```

- Pour API plus de détails, consultez la section ListCommandsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMCommandInvocation`

L'exemple de code suivant montre comment utiliser`Get-SSMCommandInvocation`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie toutes les invocations d'une commande.

```
Get-SSMCommandInvocation -CommandId "b8eac879-0541-439d-94ec-47a80d554f44" -Detail
 $true
```

Sortie :

```
CommandId          : b8eac879-0541-439d-94ec-47a80d554f44
CommandPlugins     : {aws:runShellScript}
Comment            : IP config
DocumentName       : AWS-RunShellScript
InstanceId         : i-0cb2b964d3e14fd9f
InstanceName       :
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
RequestedDateTime  : 2/22/2017 8:13:16 PM
ServiceRole        :
StandardErrorUrl   :
StandardOutputUrl  :
Status             : Success
StatusDetails      : Success
TraceOutput        :
```

Exemple 2 : Cet exemple répertorie CommandPlugins pour l'invocation de l'identifiant de commande e1eb2e3c-ed4c-5123-45c1-234f5612345f

```
Get-SSMCommandInvocation -CommandId e1eb2e3c-ed4c-5123-45c1-234f5612345f -Detail:
$true | Select-Object -ExpandProperty CommandPlugins
```

Sortie :

```
Name                  : aws:runPowerShellScript
Output                : Completed 17.7 KiB/17.7 KiB (40.1 KiB/s) with 1 file(s)
 remainingdownload: s3://dd-aess-r-ctmer/KUMO.png to ..\..\programdata\KUMO.png
                         kumo available

OutputS3BucketName    :
OutputS3KeyPrefix     :
OutputS3Region        : eu-west-1
ResponseCode          : 0
ResponseFinishDateTime : 4/3/2019 11:53:23 AM
ResponseStartDateTime  : 4/3/2019 11:53:21 AM
StandardErrorUrl      :
StandardOutputUrl     :
Status                : Success
StatusDetails         : Success
```

- Pour API plus de détails, consultez la section ListCommandInvocationsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMCommandInvocationDetail`

L'exemple de code suivant montre comment utiliser`Get-SSMCommandInvocationDetail`.

Outils pour PowerShell

Exemple 1 : Cet exemple affiche les détails d'une commande exécutée sur une instance.

```
Get-SSMCommandInvocationDetail -InstanceId "i-0cb2b964d3e14fd9f" -CommandId
 "b8eac879-0541-439d-94ec-47a80d554f44"
```

Sortie :

```
CommandId             : b8eac879-0541-439d-94ec-47a80d554f44
Comment               : IP config
DocumentName          : AWS-RunShellScript
ExecutionElapsedTime  : PT0.004S
```

```
ExecutionEndDateTime   : 2017-02-22T20:13:16.651Z
ExecutionStartDateTime : 2017-02-22T20:13:16.651Z
InstanceId             : i-0cb2b964d3e14fd9f
PluginName             : aws:runShellScript
ResponseCode           : 0
StandardErrorContent   :
StandardErrorUrl       :
StandardOutputContent  :
StandardOutputUrl      :
Status                 : Success
StatusDetails          : Success
```

- Pour API plus de détails, consultez la section GetCommandInvocationRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMComplianceItemList`

L'exemple de code suivant montre comment utiliser`Get-SSMComplianceItemList`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie la liste des éléments de conformité pour l'identifiant et le type de ressource donnés, le type de conformité de filtrage étant « Association »

```
Get-SSMComplianceItemList -ResourceId i-1a2caf345f67d0dc2 -ResourceType
 ManagedInstance -Filter @{Key="ComplianceType";Values="Association"}
```

Sortie :

```
ComplianceType   : Association
Details          : {[DocumentName, AWS-GatherSoftwareInventory], [DocumentVersion,
 1]}
ExecutionSummary : Amazon.SimpleSystemsManagement.Model.ComplianceExecutionSummary
Id               : 123a45a1-c234-1234-1245-67891236db4e
ResourceId       : i-1a2caf345f67d0dc2
ResourceType     : ManagedInstance
Severity         : UNSPECIFIED
Status           : COMPLIANT
Title            :
```

- Pour API plus de détails, consultez la section ListComplianceItemsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMComplianceSummaryList`

L'exemple de code suivant montre comment utiliser`Get-SSMComplianceSummaryList`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie un décompte récapitulatif des ressources conformes et non conformes pour tous les types de conformité.

```
Get-SSMComplianceSummaryList
```

Sortie :

```
ComplianceType CompliantSummary
 NonCompliantSummary
-------------- ----------------
 -------------------
FleetTotal     Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Association    Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Custom:InSpec  Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Patch          Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
```

- Pour API plus de détails, consultez la section ListComplianceSummariesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMConnectionStatus`

L'exemple de code suivant montre comment utiliser`Get-SSMConnectionStatus`.

Outils pour PowerShell

Exemple 1 : Cet exemple récupère l'état de connexion au gestionnaire de session d'une instance afin de déterminer si elle est connectée et prête à recevoir des connexions au gestionnaire de session.

```
Get-SSMConnectionStatus -Target i-0a1caf234f12d3dc4
```

Sortie :

```
Status    Target
------    ------
Connected i-0a1caf234f12d3dc4
```

- Pour API plus de détails, consultez la section GetConnectionStatusRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMDefaultPatchBaseline`

L'exemple de code suivant montre comment utiliser`Get-SSMDefaultPatchBaseline`.

Outils pour PowerShell

Exemple 1 : Cet exemple affiche la ligne de base de correctifs par défaut.

```
Get-SSMDefaultPatchBaseline
```

Sortie :

```
arn:aws:ssm:us-west-2:123456789012:patchbaseline/pb-04fb4ae6142167966
```

- Pour API plus de détails, consultez la section GetDefaultPatchBaselineRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMDeployablePatchSnapshotForInstance`

L'exemple de code suivant montre comment utiliser`Get-SSMDeployablePatchSnapshotForInstance`.

Outils pour PowerShell

Exemple 1 : Cet exemple affiche l'instantané actuel de la ligne de base de correctifs utilisée par une instance. Cette commande doit être exécutée depuis l'instance à l'aide des informations d'identification de l'instance. Pour s'assurer qu'il utilise les informations d'identification de l'instance, l'exemple transmet un **`Amazon.Runtime.InstanceProfileAWSCredentials`** objet au paramètre Credentials.

```
$credentials = [Amazon.Runtime.InstanceProfileAWSCredentials]::new()
Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f" -Credentials $credentials
```

Sortie :

```
InstanceId          SnapshotDownloadUrl
----------          -------------------
i-0cb2b964d3e14fd9f https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...1692/4681775b-098f-4435...
```

Exemple 2 : Cet exemple montre comment obtenir le résultat complet SnapshotDownloadUrl. Cette commande doit être exécutée depuis l'instance à l'aide des informations d'identification de l'instance. Pour s'assurer qu'elle utilise les informations d'identification de l'instance, l'exemple configure la PowerShell session pour qu'elle utilise un **`Amazon.Runtime.InstanceProfileAWSCredentials`** objet.

```
Set-AWSCredential -Credential
 ([Amazon.Runtime.InstanceProfileAWSCredentials]::new())
(Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f").SnapshotDownloadUrl
```

Sortie :

```
https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...
```

- Pour API plus de détails, consultez la section GetDeployablePatchSnapshotForInstanceRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMDocument`

L'exemple de code suivant montre comment utiliser`Get-SSMDocument`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie le contenu d'un document.

```
Get-SSMDocument -Name "RunShellScript"
```

Sortie :

```
Content
-------
{...
```

Exemple 2 : Cet exemple affiche le contenu complet d'un document.

```
(Get-SSMDocument -Name "RunShellScript").Content
{
   "schemaVersion":"2.0",
   "description":"Run an updated script",
   "parameters":{
      "commands":{
         "type":"StringList",
         "description":"(Required) Specify a shell script or a command to run.",
         "minItems":1,
         "displayType":"textarea"
      }
   },
   "mainSteps":[
      {
         "action":"aws:runShellScript",
         "name":"runShellScript",
         "inputs":{
            "commands":"{{ commands }}"
         }
      },
      {
         "action":"aws:runPowerShellScript",
         "name":"runPowerShellScript",
         "inputs":{
            "commands":"{{ commands }}"
         }
      }
   ]
}
```

- Pour API plus de détails, consultez la section GetDocumentRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMDocumentDescription`

L'exemple de code suivant montre comment utiliser`Get-SSMDocumentDescription`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie des informations relatives à un document.

```
Get-SSMDocumentDescription -Name "RunShellScript"
```

Sortie :

```
CreatedDate     : 2/24/2017 5:25:13 AM
DefaultVersion  : 1
Description     : Run an updated script
DocumentType    : Command
DocumentVersion : 1
Hash            : f775e5df4904c6fa46686c4722fae9de1950dace25cd9608ff8d622046b68d9b
HashType        : Sha256
LatestVersion   : 1
Name            : RunShellScript
Owner           : 123456789012
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion   : 2.0
Sha1            :
Status          : Active
```

- Pour API plus de détails, consultez la section DescribeDocumentRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMDocumentList`

L'exemple de code suivant montre comment utiliser`Get-SSMDocumentList`.

Outils pour PowerShell

Exemple 1 : Répertorie tous les documents de configuration de votre compte.

```
Get-SSMDocumentList
```

Sortie :

```
DocumentType    : Command
DocumentVersion : 1
Name            : AWS-ApplyPatchBaseline
Owner           : Amazon
PlatformTypes   : {Windows}
SchemaVersion   : 1.2

DocumentType    : Command
DocumentVersion : 1
Name            : AWS-ConfigureAWSPackage
Owner           : Amazon
PlatformTypes   : {Windows, Linux}
SchemaVersion   : 2.0

DocumentType    : Command
DocumentVersion : 1
Name            : AWS-ConfigureCloudWatch
Owner           : Amazon
PlatformTypes   : {Windows}
SchemaVersion   : 1.2
...
```

Exemple 2 : Cet exemple récupère tous les documents d'automatisation dont le nom correspond à « Platform »

```
Get-SSMDocumentList -DocumentFilterList @{Key="DocumentType";Value="Automation"} |
 Where-Object Name -Match "Platform"
```

Sortie :

```
DocumentFormat  : JSON
DocumentType    : Automation
DocumentVersion : 7
Name            : KT-Get-Platform
Owner           : 987654123456
PlatformTypes   : {Windows, Linux}
SchemaVersion   : 0.3
Tags            : {}
TargetType      :
VersionName     :
```

- Pour API plus de détails, consultez la section ListDocumentsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMDocumentPermission`

L'exemple de code suivant montre comment utiliser`Get-SSMDocumentPermission`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie toutes les versions d'un document.

```
Get-SSMDocumentVersionList -Name "RunShellScript"
```

Sortie :

```
CreatedDate          DocumentVersion IsDefaultVersion Name
-----------          --------------- ---------------- ----
2/24/2017 5:25:13 AM 1               True             RunShellScript
```

- Pour API plus de détails, consultez la section DescribeDocumentPermissionRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMDocumentVersionList`

L'exemple de code suivant montre comment utiliser`Get-SSMDocumentVersionList`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie la liste des autorisations pour un document.

```
Get-SSMDocumentPermission -Name "RunShellScript" -PermissionType "Share"
```

Sortie :

```
all
```

- Pour API plus de détails, consultez la section ListDocumentVersionsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMEffectiveInstanceAssociationList`

L'exemple de code suivant montre comment utiliser`Get-SSMEffectiveInstanceAssociationList`.

Outils pour PowerShell

Exemple 1 : Cet exemple décrit les associations efficaces pour une instance.

```
Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -MaxResult
 5
```

Sortie :

```
AssociationId                        Content
-------------                        -------
d8617c07-2079-4c18-9847-1655fc2698b0 {...
```

Exemple 2 : Cet exemple affiche le contenu des associations efficaces pour une instance.

```
(Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -
MaxResult 5).Content
```

Sortie :

```
{
    "schemaVersion": "1.2",
    "description": "Update the Amazon SSM Agent to the latest version or specified
 version.",
    "parameters": {
        "version": {
            "default": "",
            "description": "(Optional) A specific version of the Amazon SSM Agent to
 install. If not specified, the agen
t will be updated to the latest version.",
            "type": "String"
        },
        "allowDowngrade": {
            "default": "false",
            "description": "(Optional) Allow the Amazon SSM Agent service to be
 downgraded to an earlier version. If set
```

```
 to false, the service can be upgraded to newer versions only (default). If set to
 true, specify the earlier version.",
            "type": "String",
            "allowedValues": [
                "true",
                "false"
            ]
        }
    },
    "runtimeConfig": {
        "aws:updateSsmAgent": {
            "properties": [
                {
                "agentName": "amazon-ssm-agent",
                "source": "https://s3.{Region}.amazonaws.com/amazon-ssm-{Region}/
ssm-agent-manifest.json",
                "allowDowngrade": "{{ allowDowngrade }}",
                "targetVersion": "{{ version }}"
                }
            ]
        }
    }
}
```

- Pour API plus de détails, consultez la section DescribeEffectiveInstanceAssociationsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMEffectivePatchesForPatchBaseline`

L'exemple de code suivant montre comment utiliser`Get-SSMEffectivePatchesForPatchBaseline`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie toutes les lignes de base des correctifs, avec une liste de résultats maximale de 1.

```
Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1
```

Sortie :

```
Patch                                       PatchStatus
-----                                       -----------
Amazon.SimpleSystemsManagement.Model.Patch
 Amazon.SimpleSystemsManagement.Model.PatchStatus
```

Exemple 2 : Cet exemple affiche l'état du correctif pour toutes les lignes de base des correctifs, avec une liste de résultats maximale de 1.

```
(Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1).PatchStatus
```

Sortie :

```
ApprovalDate          DeploymentStatus
------------          ----------------
12/21/2010 6:00:00 PM APPROVED
```

- Pour API plus de détails, consultez la section DescribeEffectivePatchesForPatchBaselineRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMInstanceAssociationsStatus`

L'exemple de code suivant montre comment utiliser`Get-SSMInstanceAssociationsStatus`.

Outils pour PowerShell

Exemple 1 : Cet exemple montre les détails des associations associées à une instance.

```
Get-SSMInstanceAssociationsStatus -InstanceId "i-0000293ffd8c57862"
```

Sortie :

```
AssociationId    : d8617c07-2079-4c18-9847-1655fc2698b0
DetailedStatus   : Pending
DocumentVersion  : 1
ErrorCode        :
ExecutionDate    : 2/20/2015 8:31:11 AM
ExecutionSummary : temp_status_change
InstanceId       : i-0000293ffd8c57862
```

```
Name             : AWS-UpdateSSMAgent
OutputUrl        :
Status           : Pending
```

Exemple 2 : Cet exemple vérifie le statut de l'association d'instance pour l'identifiant d'instance donné et affiche ensuite le statut d'exécution de ces associations

```
Get-SSMInstanceAssociationsStatus -InstanceId i-012e3cb4df567e8aa | ForEach-Object
 {Get-SSMAssociationExecution -AssociationId .AssociationId}
```

Sortie :

```
AssociationId         : 512a34a5-c678-1234-1234-12345678db9e
AssociationVersion    : 2
CreatedTime           : 3/2/2019 8:53:29 AM
DetailedStatus        :
ExecutionId           : 512a34a5-c678-1234-1234-12345678db9e
LastExecutionDate     : 1/1/0001 12:00:00 AM
ResourceCountByStatus : {Success=9}
Status                : Success
```

- Pour API plus de détails, consultez la section DescribeInstanceAssociationsStatusRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMInstanceInformation`

L'exemple de code suivant montre comment utiliser`Get-SSMInstanceInformation`.

Outils pour PowerShell

Exemple 1 : Cet exemple montre les détails de chacune de vos instances.

```
Get-SSMInstanceInformation
```

Sortie :

```
ActivationId                          :
AgentVersion                          : 2.0.672.0
AssociationOverview                   :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
```

```
AssociationStatus                       : Success
ComputerName                            : ip-172-31-44-222.us-west-2.compute.internal
IamRole                                 :
InstanceId                              : i-0cb2b964d3e14fd9f
IPAddress                               : 172.31.44.222
IsLatestVersion                         : True
LastAssociationExecutionDate            : 2/24/2017 3:18:09 AM
LastPingDateTime                        : 2/24/2017 3:35:03 AM
LastSuccessfulAssociationExecutionDate : 2/24/2017 3:18:09 AM
Name                                    :
PingStatus                              : ConnectionLost
PlatformName                            : Amazon Linux AMI
PlatformType                            : Linux
PlatformVersion                         : 2016.09
RegistrationDate                        : 1/1/0001 12:00:00 AM
ResourceType                            : EC2Instance
```

Exemple 2 : Cet exemple montre comment utiliser le paramètre -Filter pour filtrer les résultats uniquement en fonction des instances de AWS Systems Manager situées dans une région **`us-east-1`** avec un **`AgentVersion`** de**`2.2.800.0`**. Vous trouverez une liste des valeurs de clé -Filter valides dans la rubrique de InstanceInformation API référence (https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformation.html #systemsmanager -Type- -). InstanceInformation ActivationId

```
$Filters = @{
    Key="AgentVersion"
    Values="2.2.800.0"
}
Get-SSMInstanceInformation -Region us-east-1 -Filter $Filters
```

Sortie :

```
ActivationId                            :
AgentVersion                            : 2.2.800.0
AssociationOverview                     :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                       : Success
ComputerName                            : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                 :
InstanceId                              : i-EXAMPLEb0792d98ce
IPAddress                               : 10.0.0.01
IsLatestVersion                         : False
```

```
LastAssociationExecutionDate           : 8/16/2018 12:02:50 AM
LastPingDateTime                       : 8/16/2018 7:40:27 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:02:50 AM
Name                                   :
PingStatus                             : Online
PlatformName                           : Microsoft Windows Server 2016 Datacenter
PlatformType                           : Windows
PlatformVersion                        : 10.0.14393
RegistrationDate                       : 1/1/0001 12:00:00 AM
ResourceType                           : EC2Instance

ActivationId                           :
AgentVersion                           : 2.2.800.0
AssociationOverview                    :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                      : Success
ComputerName                           : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                :
InstanceId                             : i-EXAMPLEac7501d023
IPAddress                              : 10.0.0.02
IsLatestVersion                        : False
LastAssociationExecutionDate           : 8/16/2018 12:00:20 AM
LastPingDateTime                       : 8/16/2018 7:40:35 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:00:20 AM
Name                                   :
PingStatus                             : Online
PlatformName                           : Microsoft Windows Server 2016 Datacenter
PlatformType                           : Windows
PlatformVersion                        : 10.0.14393
RegistrationDate                       : 1/1/0001 12:00:00 AM
ResourceType                           : EC2Instance
```

Exemple 3 : Cet exemple montre comment utiliser le InstanceInformationFilterList paramètre - pour filtrer les résultats uniquement en fonction des instances **`PlatformTypes`** de AWS Systems Manager situées dans **`us-east-1`** une région avec **`Windows`** ou**`Linux`**. Vous trouverez une liste des valeurs InstanceInformationFilterList clés valides dans la rubrique de InstanceInformationFilter API référence (https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformationFilter.html).

```
$Filters = @{
   Key="PlatformTypes"
   ValueSet=("Windows","Linux")
```

```
}
Get-SSMInstanceInformation -Region us-east-1 -InstanceInformationFilterList $Filters
```

Sortie :

```
ActivationId                          :
AgentVersion                          : 2.2.800.0
AssociationOverview                   :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                     : Success
ComputerName                          : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                               :
InstanceId                            : i-EXAMPLEb0792d98ce
IPAddress                             : 10.0.0.27
IsLatestVersion                       : False
LastAssociationExecutionDate          : 8/16/2018 12:02:50 AM
LastPingDateTime                      : 8/16/2018 7:40:27 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:02:50 AM
Name                                  :
PingStatus                            : Online
PlatformName                          : Ubuntu Server 18.04 LTS
PlatformType                          : Linux
PlatformVersion                       : 18.04
RegistrationDate                      : 1/1/0001 12:00:00 AM
ResourceType                          : EC2Instance

ActivationId                          :
AgentVersion                          : 2.2.800.0
AssociationOverview                   :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                     : Success
ComputerName                          : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                               :
InstanceId                            : i-EXAMPLEac7501d023
IPAddress                             : 10.0.0.100
IsLatestVersion                       : False
LastAssociationExecutionDate          : 8/16/2018 12:00:20 AM
LastPingDateTime                      : 8/16/2018 7:40:35 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:00:20 AM
Name                                  :
PingStatus                            : Online
PlatformName                          : Microsoft Windows Server 2016 Datacenter
PlatformType                          : Windows
```

```
PlatformVersion                 : 10.0.14393
RegistrationDate                : 1/1/0001 12:00:00 AM
ResourceType                    : EC2Instance
```

Exemple 4 : Cet exemple répertorie les instances gérées par SSM et les exportations InstanceId PingStatus, LastPingDateTime et PlatformName vers un fichier csv.

```
Get-SSMInstanceInformation | Select-Object InstanceId, PingStatus, LastPingDateTime,
 PlatformName | Export-Csv Instance-details.csv -NoTypeInformation
```

- Pour API plus de détails, consultez la section DescribeInstanceInformationRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMInstancePatch`

L'exemple de code suivant montre comment utiliser`Get-SSMInstancePatch`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir les détails de conformité des correctifs pour une instance.

```
Get-SSMInstancePatch -InstanceId "i-08ee91c0b17045407"
```

- Pour API plus de détails, consultez la section DescribeInstancePatchesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMInstancePatchState`

L'exemple de code suivant montre comment utiliser`Get-SSMInstancePatchState`.

Outils pour PowerShell

Exemple 1 : Cet exemple obtient les états récapitulatifs des correctifs pour une instance.

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407"
```

Exemple 2 : Cet exemple obtient les états récapitulatifs des correctifs pour deux instances.

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407","i-09a618aec652973a9"
```

- Pour API plus de détails, consultez la section DescribeInstancePatchStatesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMInstancePatchStatesForPatchGroup`

L'exemple de code suivant montre comment utiliser`Get-SSMInstancePatchStatesForPatchGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple obtient les états récapitulatifs des correctifs par instance pour un groupe de correctifs.

```
Get-SSMInstancePatchStatesForPatchGroup -PatchGroup "Production"
```

- Pour API plus de détails, consultez la section DescribeInstancePatchStatesForPatchGroupRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMInventory`

L'exemple de code suivant montre comment utiliser`Get-SSMInventory`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir les métadonnées personnalisées de votre inventaire.

```
Get-SSMInventory
```

Sortie :

```
Data
  Id
----
  --
{[AWS:InstanceInformation,
 Amazon.SimpleSystemsManagement.Model.InventoryResultItem]} i-0cb2b964d3e14fd9f
```

- Pour API plus de détails, consultez la section GetInventoryRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMInventoryEntriesList`

L'exemple de code suivant montre comment utiliser`Get-SSMInventoryEntriesList`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie toutes les entrées d'inventaire personnalisées pour une instance.

```
Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
 "Custom:RackInfo"
```

Sortie :

```
CaptureTime   : 2016-08-22T10:01:01Z
Entries       :
 {Amazon.Runtime.Internal.Util.AlwaysSendDictionary`2[System.String,System.String]}
InstanceId    : i-0cb2b964d3e14fd9f
NextToken     :
SchemaVersion : 1.0
TypeName      : Custom:RackInfo
```

Exemple 2 : Cet exemple répertorie les détails.

```
(Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
 "Custom:RackInfo").Entries
```

Sortie :

```
Key          Value
---          -----
RackLocation Bay B/Row C/Rack D/Shelf E
```

- Pour API plus de détails, consultez la section ListInventoryEntriesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMInventoryEntryList`

L'exemple de code suivant montre comment utiliser`Get-SSMInventoryEntryList`.

Outils pour PowerShell

Exemple 1 : Cet exemple récupère les entrées **AWS:Network** d'inventaire des types pour l'instance.

```
Get-SSMInventoryEntryList -InstanceId mi-088dcb0ecea37b076 -TypeName AWS:Network |
 Select-Object -ExpandProperty Entries
```

Sortie :

```
Key        Value
---        -----
DHCPServer 172.31.11.2
DNSServer  172.31.0.1
Gateway    172.31.11.2
IPV4       172.31.11.222
IPV6       fe12::3456:7da8:901a:12a3
MacAddress 1A:23:4E:5B:FB:67
Name       Amazon Elastic Network Adapter
SubnetMask 255.255.240.0
```

- Pour API plus de détails, consultez la section Référence des AWS Tools for PowerShell applets de commande Get- SSMInventoryEntryList in.

### `Get-SSMInventorySchema`

L'exemple de code suivant montre comment utiliser`Get-SSMInventorySchema`.

Outils pour PowerShell

Exemple 1 : Cet exemple renvoie une liste de noms de types d'inventaire pour le compte.

```
Get-SSMInventorySchema
```

- Pour API plus de détails, consultez la section GetInventorySchemaRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMLatestEC2Image`

L'exemple de code suivant montre comment utiliser`Get-SSMLatestEC2Image`.

Exemple 1 : Cet exemple répertorie toutes les dernières versions de WindowsAMIs.

```
PS Get-SSMLatestEC2Image -Path ami-windows-latest
```

Sortie :

```
Name                                                                           Value
----                                                                           -----
Windows_Server-2008-R2_SP1-English-64Bit-SQL_2012_SP4_Express
 ami-0e5ddd288daff4fab
Windows_Server-2012-R2_RTM-Chinese_Simplified-64Bit-Base
 ami-0c5ea64e6bec1cb50
Windows_Server-2012-R2_RTM-Chinese_Traditional-64Bit-Base
 ami-09775eff0bf8c113d
Windows_Server-2012-R2_RTM-Dutch-64Bit-Base
 ami-025064b67e28cf5df
...
```

Exemple 2 : Cet exemple récupère l'AMIidentifiant d'une image Amazon Linux spécifique pour la région us-west-2.

```
PS Get-SSMLatestEC2Image -Path ami-amazon-linux-latest -ImageName amzn-ami-hvm-
x86_64-ebs -Region us-west-2
```

Sortie :

```
ami-09b92cd132204c704
```

Exemple 3 : Cet exemple répertorie toutes les dernières fenêtres AMIs correspondant à l'expression générique spécifiée.

```
Get-SSMLatestEC2Image -Path ami-windows-latest -ImageName *Windows*2019*English*
```

Sortie :

```
Name                                                Value
----                                                -----
Windows_Server-2019-English-Full-SQL_2017_Web       ami-085e9d27da5b73a42
```

```
Windows_Server-2019-English-STIG-Core                    ami-0bfd85c29148c7f80
Windows_Server-2019-English-Full-SQL_2019_Web            ami-02099560d7fb11f20
Windows_Server-2019-English-Full-SQL_2016_SP2_Standard   ami-0d7ae2d81c07bd598
...
```

- Pour API plus de détails, consultez la section Référence des AWS Tools for PowerShell applets de commande Get- SSMLatestEC2Image in.

### `Get-SSMMaintenanceWindow`

L'exemple de code suivant montre comment utiliser`Get-SSMMaintenanceWindow`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir des informations sur une fenêtre de maintenance.

```
Get-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d"
```

Sortie :

```
AllowUnassociatedTargets : False
CreatedDate              : 2/20/2017 6:14:05 PM
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
ModifiedDate             : 2/20/2017 6:14:05 PM
Name                     : TestMaintWin
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

- Pour API plus de détails, consultez la section GetMaintenanceWindowRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMMaintenanceWindowExecution`

L'exemple de code suivant montre comment utiliser`Get-SSMMaintenanceWindowExecution`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie les informations relatives à une tâche exécutée dans le cadre de l'exécution d'une fenêtre de maintenance.

```
Get-SSMMaintenanceWindowExecution -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"
```

Sortie :

```
EndTime           : 2/21/2017 4:00:35 PM
StartTime         : 2/21/2017 4:00:34 PM
Status            : FAILED
StatusDetails     : One or more tasks in the orchestration failed.
TaskIds           : {ac0c6ae1-daa3-4a89-832e-d384503b6586}
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- Pour API plus de détails, consultez la section GetMaintenanceWindowExecutionRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMMaintenanceWindowExecutionList`

L'exemple de code suivant montre comment utiliser`Get-SSMMaintenanceWindowExecutionList`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie toutes les exécutions pour une fenêtre de maintenance.

```
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d"
```

Sortie :

```
EndTime           : 2/20/2017 6:30:17 PM
StartTime         : 2/20/2017 6:30:16 PM
Status            : FAILED
StatusDetails     : One or more tasks in the orchestration failed.
WindowExecutionId : 6f3215cf-4101-4fa0-9b7b-9523269599c7
WindowId          : mw-03eb9db42890fb82d
```

Exemple 2 : Cet exemple répertorie toutes les exécutions pour une fenêtre de maintenance avant une date spécifiée.

```
$option1 = @{Key="ExecutedBefore";Values=@("2016-11-04T05:00:00Z")}
```

```
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
 $option1
```

Exemple 3 : Cet exemple répertorie toutes les exécutions pour une fenêtre de maintenance après une date spécifiée.

```
$option1 = @{Key="ExecutedAfter";Values=@("2016-11-04T05:00:00Z")}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
 $option1
```

- Pour API plus de détails, consultez la section DescribeMaintenanceWindowExecutionsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMMaintenanceWindowExecutionTask`

L'exemple de code suivant montre comment utiliser`Get-SSMMaintenanceWindowExecutionTask`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie les informations relatives à une tâche qui faisait partie de l'exécution d'une fenêtre de maintenance.

```
Get-SSMMaintenanceWindowExecutionTask -TaskId "ac0c6ae1-daa3-4a89-832e-d384503b6586"
 -WindowExecutionId "518d5565-5969-4cca-8f0e-da3b2a638355"
```

Sortie :

```
EndTime           : 2/21/2017 4:00:35 PM
MaxConcurrency    : 1
MaxErrors         : 1
Priority          : 10
ServiceRole       : arn:aws:iam::123456789012:role/MaintenanceWindowsRole
StartTime         : 2/21/2017 4:00:34 PM
Status            : FAILED
StatusDetails     : The maximum error count was exceeded.
TaskArn           : AWS-RunShellScript
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
TaskParameters    :
 {Amazon.Runtime.Internal.Util.AlwaysSendDictionary`2[System.String,Amazon.SimpleSystemsMana
```

```
                    meterValueExpression]}
Type              : RUN_COMMAND
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- Pour API plus de détails, consultez la section GetMaintenanceWindowExecutionTaskRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMMaintenanceWindowExecutionTaskInvocationList`

L'exemple de code suivant montre comment utiliser`Get-SSMMaintenanceWindowExecutionTaskInvocationList`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie les appels pour une tâche exécutée dans le cadre de l'exécution d'une fenêtre de maintenance.

```
Get-SSMMaintenanceWindowExecutionTaskInvocationList -TaskId "ac0c6ae1-
daa3-4a89-832e-d384503b6586" -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"
```

Sortie :

```
EndTime           : 2/21/2017 4:00:34 PM
ExecutionId       :
InvocationId      : e274b6e1-fe56-4e32-bd2a-8073c6381d8b
OwnerInformation  :
Parameters        : {"documentName":"AWS-RunShellScript","instanceIds":
["i-0000293ffd8c57862"],"parameters":{"commands":["df"]},"maxConcurrency":"1",
                    "maxErrors":"1"}
StartTime         : 2/21/2017 4:00:34 PM
Status            : FAILED
StatusDetails     : The instance IDs list contains an invalid entry.
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
WindowTargetId    :
```

- Pour API plus de détails, consultez la section DescribeMaintenanceWindowExecutionTaskInvocationsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMMaintenanceWindowExecutionTaskList`

L'exemple de code suivant montre comment utiliser`Get-SSMMaintenanceWindowExecutionTaskList`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie les tâches associées à l'exécution d'une fenêtre de maintenance.

```
Get-SSMMaintenanceWindowExecutionTaskList -WindowExecutionId
 "518d5565-5969-4cca-8f0e-da3b2a638355"
```

Sortie :

```
EndTime           : 2/21/2017 4:00:35 PM
StartTime         : 2/21/2017 4:00:34 PM
Status            : SUCCESS
TaskArn           : AWS-RunShellScript
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
TaskType          : RUN_COMMAND
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- Pour API plus de détails, consultez la section DescribeMaintenanceWindowExecutionTasksRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMMaintenanceWindowList`

L'exemple de code suivant montre comment utiliser`Get-SSMMaintenanceWindowList`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie toutes les fenêtres de maintenance de votre compte.

```
Get-SSMMaintenanceWindowList
```

Sortie :

```
Cutoff   : 1
Duration : 4
```

```
Enabled  : True
Name     : My-First-Maintenance-Window
WindowId : mw-06d59c1a07c022145
```

- Pour API plus de détails, consultez la section DescribeMaintenanceWindowsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMMaintenanceWindowTarget`

L'exemple de code suivant montre comment utiliser`Get-SSMMaintenanceWindowTarget`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie toutes les cibles d'une fenêtre de maintenance.

```
Get-SSMMaintenanceWindowTarget -WindowId "mw-06cf17cbefcb4bf4f"
```

Sortie :

```
OwnerInformation : Single instance
ResourceType     : INSTANCE
Targets          : {InstanceIds}
WindowId         : mw-06cf17cbefcb4bf4f
WindowTargetId   : 350d44e6-28cc-44e2-951f-4b2c985838f6

OwnerInformation : Two instances in a list
ResourceType     : INSTANCE
Targets          : {InstanceIds}
WindowId         : mw-06cf17cbefcb4bf4f
WindowTargetId   : e078a987-2866-47be-bedd-d9cf49177d3a
```

- Pour API plus de détails, consultez la section DescribeMaintenanceWindowTargetsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMMaintenanceWindowTaskList`

L'exemple de code suivant montre comment utiliser`Get-SSMMaintenanceWindowTaskList`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie toutes les tâches d'une fenêtre de maintenance.

```
Get-SSMMaintenanceWindowTaskList -WindowId "mw-06cf17cbefcb4bf4f"
```

Sortie :

```
LoggingInfo    :
MaxConcurrency : 1
MaxErrors      : 1
Priority       : 10
ServiceRoleArn : arn:aws:iam::123456789012:role/MaintenanceWindowsRole
Targets        : {InstanceIds}
TaskArn        : AWS-RunShellScript
TaskParameters : {[commands,
 Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression]}
Type           : RUN_COMMAND
WindowId       : mw-06cf17cbefcb4bf4f
WindowTaskId   : a23e338d-ff30-4398-8aa3-09cd052ebf17
```

- Pour API plus de détails, consultez la section DescribeMaintenanceWindowTasksRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMParameterHistory`

L'exemple de code suivant montre comment utiliser`Get-SSMParameterHistory`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie l'historique des valeurs d'un paramètre.

```
Get-SSMParameterHistory -Name "Welcome"
```

Sortie :

```
Description      :
KeyId            :
LastModifiedDate : 3/3/2017 6:55:25 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type             : String
Value            : helloWorld
```

- Pour API plus de détails, consultez la section GetParameterHistoryRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMParameterList`

L'exemple de code suivant montre comment utiliser`Get-SSMParameterList`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie tous les paramètres.

```
Get-SSMParameterList
```

Sortie :

```
Description      :
KeyId            :
LastModifiedDate : 3/3/2017 6:58:23 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type             : String
```

- Pour API plus de détails, consultez la section DescribeParametersRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMParameterValue`

L'exemple de code suivant montre comment utiliser`Get-SSMParameterValue`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie les valeurs d'un paramètre.

```
Get-SSMParameterValue -Name "Welcome"
```

Sortie :

```
InvalidParameters Parameters
----------------- ----------
{}                {Welcome}
```

Exemple 2 : Cet exemple répertorie les détails de la valeur.

```
(Get-SSMParameterValue -Name "Welcome").Parameters
```

Sortie :

```
Name    Type   Value
----    ----   -----
Welcome String Good day, Sunshine!
```

- Pour API plus de détails, consultez la section GetParametersRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMPatchBaseline`

L'exemple de code suivant montre comment utiliser`Get-SSMPatchBaseline`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie toutes les lignes de base des correctifs.

```
Get-SSMPatchBaseline
```

Sortie :

```
BaselineDescription                                                 BaselineId
                                                 BaselineName
------------------                                                  ----------
                                                 ------------
Default Patch Baseline Provided by AWS.                             arn:aws:ssm:us-
west-2:123456789012:patchbaseline/pb-04fb4ae6142167966 AWS-DefaultP...
Baseline containing all updates approved for production systems pb-045f10b4f382baeda
                                                 Production-B...
Baseline containing all updates approved for production systems pb-0a2f1059b670ebd31
                                                 Production-B...
```

Exemple 2 : Cet exemple répertorie toutes les lignes de base de correctifs fournies par AWS. La syntaxe utilisée dans cet exemple nécessite PowerShell la version 3 ou ultérieure.

```
$filter1 = @{Key="OWNER";Values=@("AWS")}
```

Sortie :

```
Get-SSMPatchBaseline -Filter $filter1
```

Exemple 3 : Cet exemple répertorie toutes les lignes de base des correctifs dont vous êtes le propriétaire. La syntaxe utilisée dans cet exemple nécessite PowerShell la version 3 ou ultérieure.

```
$filter1 = @{Key="OWNER";Values=@("Self")}
```

Sortie :

```
Get-SSMPatchBaseline -Filter $filter1
```

Exemple 4 : Avec PowerShell la version 2, vous devez utiliser New-Object pour créer chaque balise.

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "OWNER"
$filter1.Values = "AWS"

Get-SSMPatchBaseline -Filter $filter1
```

Sortie :

```
BaselineDescription                     BaselineId
                          BaselineName             DefaultBaselin

                                                   e
-------------------                     ----------
                          ------------             --------------
Default Patch Baseline Provided by AWS. arn:aws:ssm:us-
west-2:123456789012:patchbaseline/pb-04fb4ae6142167966 AWS-DefaultPatchBaseline True
```

- Pour API plus de détails, consultez la section DescribePatchBaselinesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMPatchBaselineDetail`

L'exemple de code suivant montre comment utiliser`Get-SSMPatchBaselineDetail`.

Outils pour PowerShell

Exemple 1 : Cet exemple affiche les détails d'une ligne de base de correctifs.

```
Get-SSMPatchBaselineDetail -BaselineId "pb-03da896ca3b68b639"
```

Sortie :

```
ApprovalRules   : Amazon.SimpleSystemsManagement.Model.PatchRuleGroup
ApprovedPatches : {}
BaselineId      : pb-03da896ca3b68b639
CreatedDate     : 3/3/2017 5:02:19 PM
Description     : Baseline containing all updates approved for production systems
GlobalFilters   : Amazon.SimpleSystemsManagement.Model.PatchFilterGroup
ModifiedDate    : 3/3/2017 5:02:19 PM
Name            : Production-Baseline
PatchGroups     : {}
RejectedPatches : {}
```

- Pour API plus de détails, consultez la section GetPatchBaselineRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMPatchBaselineForPatchGroup`

L'exemple de code suivant montre comment utiliser`Get-SSMPatchBaselineForPatchGroup`.

Outils pour PowerShell

Exemple 1 : cet exemple affiche la ligne de base de correctifs pour un groupe de correctifs.

```
Get-SSMPatchBaselineForPatchGroup -PatchGroup "Production"
```

Sortie :

```
BaselineId           PatchGroup
----------           ----------
pb-045f10b4f382baeda Production
```

- Pour API plus de détails, consultez la section GetPatchBaselineForPatchGroupRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMPatchGroup`

L'exemple de code suivant montre comment utiliser`Get-SSMPatchGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie les enregistrements de groupes de correctifs.

```
Get-SSMPatchGroup
```

Sortie :

```
BaselineIdentity                                          PatchGroup
----------------                                          ----------
Amazon.SimpleSystemsManagement.Model.PatchBaselineIdentity Production
```

- Pour API plus de détails, consultez la section DescribePatchGroupsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMPatchGroupState`

L'exemple de code suivant montre comment utiliser`Get-SSMPatchGroupState`.

Outils pour PowerShell

Exemple 1 : Cet exemple permet d'obtenir le résumé de haut niveau de conformité des correctifs pour un groupe de correctifs.

```
Get-SSMPatchGroupState -PatchGroup "Production"
```

Sortie :

```
Instances                     : 4
InstancesWithFailedPatches    : 1
InstancesWithInstalledOtherPatches : 4
InstancesWithInstalledPatches : 3
InstancesWithMissingPatches   : 0
InstancesWithNotApplicablePatches  : 0
```

- Pour API plus de détails, consultez la section DescribePatchGroupStateRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMResourceComplianceSummaryList`

L'exemple de code suivant montre comment utiliser`Get-SSMResourceComplianceSummaryList`.

Outils pour PowerShell

Exemple 1 : Cet exemple obtient un décompte récapitulatif au niveau des ressources. Le résumé inclut des informations sur les statuts de conformité et de non-conformité ainsi que le nombre détaillé de gravité des éléments de conformité pour les produits correspondant à « Windows10 ». Comme la valeur MaxResult par défaut est 100 si le paramètre n'est pas spécifié et que cette valeur n'est pas valide, le MaxResult paramètre est ajouté et la valeur est définie sur 50.

```
$FilterValues = @{
  "Key"="Product"
        "Type"="EQUAL"
        "Values"="Windows10"
}
        Get-SSMResourceComplianceSummaryList -Filter $FilterValues -MaxResult 50
```

- Pour API plus de détails, consultez la section ListResourceComplianceSummariesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-SSMResourceTag`

L'exemple de code suivant montre comment utiliser`Get-SSMResourceTag`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie les balises d'une fenêtre de maintenance.

```
Get-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow"
```

Sortie :

```
Key   Value
---   -----
Stack Production
```

- Pour API plus de détails, consultez la section ListTagsForResourceRéférence des AWS Tools for PowerShell applets de commande.

### `New-SSMActivation`

L'exemple de code suivant montre comment utiliser`New-SSMActivation`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une instance gérée.

```
New-SSMActivation -DefaultInstanceName "MyWebServers" -IamRole "SSMAutomationRole" -
RegistrationLimit 10
```

Sortie :

```
ActivationCode       ActivationId
--------------       ------------
KWChhOxBTiwDcKE9BlKC 08e51e79-1e36-446c-8e63-9458569c1363
```

- Pour API plus de détails, consultez la section CreateActivationRéférence des AWS Tools for PowerShell applets de commande.

### `New-SSMAssociation`

L'exemple de code suivant montre comment utiliser`New-SSMAssociation`.

Outils pour PowerShell

Exemple 1 : Cet exemple associe un document de configuration à une instance, en utilisant l'instanceIDs.

```
New-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

Sortie :

```
Name                 : AWS-UpdateSSMAgent
InstanceId           : i-0000293ffd8c57862
Date                 : 2/23/2017 6:55:22 PM
Status.Name          : Associated
Status.Date          : 2/20/2015 8:31:11 AM
Status.Message       : Associated with AWS-UpdateSSMAgent
```

```
Status.AdditionalInfo :
```

Exemple 2 : Cet exemple associe un document de configuration à une instance, en utilisant des cibles.

```
$target = @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
New-SSMAssociation -Name "AWS-UpdateSSMAgent" -Target $target
```

Sortie :

```
Name                 : AWS-UpdateSSMAgent
InstanceId           :
Date                 : 3/1/2017 6:22:21 PM
Status.Name          :
Status.Date          :
Status.Message       :
Status.AdditionalInfo :
```

Exemple 3 : Cet exemple associe un document de configuration à une instance, à l'aide de cibles et de paramètres.

```
$target = @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
$params = @{
  "action"="configure"
  "mode"="ec2"
  "optionalConfigurationSource"="ssm"
  "optionalConfigurationLocation"=""
  "optionalRestart"="yes"
}
New-SSMAssociation -Name "Configure-CloudWatch" -AssociationName "CWConfiguration" -
Target $target -Parameter $params
```

Sortie :

```
Name                 : Configure-CloudWatch
InstanceId           :
Date                 : 5/17/2018 3:17:44 PM
Status.Name          :
Status.Date          :
Status.Message       :
Status.AdditionalInfo :
```

Exemple 4 : Cet exemple crée une association avec toutes les instances de la région, avec**`AWS-GatherSoftwareInventory`**. Il fournit également des fichiers personnalisés et des emplacements de registre dans les paramètres à collecter

```
$params =
 [Collections.Generic.Dictionary[String,Collections.Generic.List[String]]]::new()
$params["windowsRegistry"] ='[{"Path":"HKEY_LOCAL_MACHINE\SOFTWARE\Amazon
\MachineImage","Recursive":false,"ValueNames":["AMIName"]}]'
$params["files"] = '[{"Path":"C:\Program Files","Pattern":
["*.exe"],"Recursive":true}, {"Path":"C:\ProgramData","Pattern":
["*.log"],"Recursive":true}]'
New-SSMAssociation -AssociationName new-in-mum -Name AWS-GatherSoftwareInventory
 -Target @{Key="instanceids";Values="*"} -Parameter $params -region ap-south-1 -
ScheduleExpression "rate(720 minutes)"
```

Sortie :

```
Name                  : AWS-GatherSoftwareInventory
InstanceId            :
Date                  : 6/9/2019 8:57:56 AM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

- Pour API plus de détails, consultez la section CreateAssociationRéférence des AWS Tools for PowerShell applets de commande.

### `New-SSMAssociationFromBatch`

L'exemple de code suivant montre comment utiliser`New-SSMAssociationFromBatch`.

Outils pour PowerShell

Exemple 1 : Cet exemple associe un document de configuration à plusieurs instances. La sortie renvoie une liste des opérations réussies et échouées, le cas échéant.

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@("AWS-UpdateSSMAgent")}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@("AWS-UpdateSSMAgent")}
New-SSMAssociationFromBatch -Entry $option1,$option2
```

Sortie :

```
Failed  Successful
------  ----------
{}      {Amazon.SimpleSystemsManagement.Model.FailedCreateAssociation,
 Amazon.SimpleSystemsManagement.Model.FailedCreateAsso...
```

Exemple 2 : Cet exemple montre tous les détails d'une opération réussie.

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@("AWS-UpdateSSMAgent")}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@("AWS-UpdateSSMAgent")}
(New-SSMAssociationFromBatch -Entry $option1,$option2).Successful
```

- Pour API plus de détails, consultez la section CreateAssociationBatchRéférence des AWS Tools for PowerShell applets de commande.

### `New-SSMDocument`

L'exemple de code suivant montre comment utiliser`New-SSMDocument`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un document dans votre compte. Le document doit être au JSON format. Pour plus d'informations sur la rédaction d'un document de configuration, voir Document de configuration dans la SSM API référence.

```
New-SSMDocument -Content (Get-Content -Raw "c:\temp\RunShellScript.json") -Name
 "RunShellScript" -DocumentType "Command"
```

Sortie :

```
CreatedDate     : 3/1/2017 1:21:33 AM
DefaultVersion  : 1
Description     : Run an updated script
DocumentType    : Command
DocumentVersion : 1
Hash            : 1d5ce820e999ff051eb4841ed887593daf77120fd76cae0d18a53cc42e4e22c1
HashType        : Sha256
LatestVersion   : 1
Name            : RunShellScript
```

```
Owner           : 809632081692
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion   : 2.0
Sha1            :
Status          : Creating
```

- Pour API plus de détails, consultez la section CreateDocumentRéférence des AWS Tools for PowerShell applets de commande.

### `New-SSMMaintenanceWindow`

L'exemple de code suivant montre comment utiliser`New-SSMMaintenanceWindow`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une nouvelle fenêtre de maintenance portant le nom spécifié qui s'exécute à 16 heures tous les mardis pendant 4 heures, avec une limite d'une heure, et qui autorise des cibles non associées.

```
New-SSMMaintenanceWindow -Name "MyMaintenanceWindow" -Duration 4 -Cutoff 1 -
AllowUnassociatedTarget $true -Schedule "cron(0 16 ? * TUE *)"
```

Sortie :

```
mw-03eb53e1ea7383998
```

- Pour API plus de détails, consultez la section CreateMaintenanceWindowRéférence des AWS Tools for PowerShell applets de commande.

### `New-SSMPatchBaseline`

L'exemple de code suivant montre comment utiliser`New-SSMPatchBaseline`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée une ligne de base de correctifs qui approuve les correctifs, sept jours après leur publication par Microsoft, pour les instances gérées exécutant Windows Server 2019 dans un environnement de production.

```
$rule = New-Object Amazon.SimpleSystemsManagement.Model.PatchRule
$rule.ApproveAfterDays = 7

$ruleFilters = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilterGroup

$patchFilter = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilter
$patchFilter.Key="PRODUCT"
$patchFilter.Values="WindowsServer2019"

$severityFilter = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilter
$severityFilter.Key="MSRC_SEVERITY"
$severityFilter.Values.Add("Critical")
$severityFilter.Values.Add("Important")
$severityFilter.Values.Add("Moderate")

$classificationFilter = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilter
$classificationFilter.Key = "CLASSIFICATION"
$classificationFilter.Values.Add( "SecurityUpdates" )
$classificationFilter.Values.Add( "Updates" )
$classificationFilter.Values.Add( "UpdateRollups" )
$classificationFilter.Values.Add( "CriticalUpdates" )

$ruleFilters.PatchFilters.Add($severityFilter)
$ruleFilters.PatchFilters.Add($classificationFilter)
$ruleFilters.PatchFilters.Add($patchFilter)
$rule.PatchFilterGroup = $ruleFilters

New-SSMPatchBaseline -Name "Production-Baseline-Windows2019" -Description "Baseline
 containing all updates approved for production systems" -ApprovalRules_PatchRule
 $rule
```

Sortie :

```
pb-0z4z6221c4296b23z
```

- Pour API plus de détails, consultez la section CreatePatchBaselineRéférence des AWS Tools for PowerShell applets de commande.

### `Register-SSMDefaultPatchBaseline`

L'exemple de code suivant montre comment utiliser`Register-SSMDefaultPatchBaseline`.

Outils pour PowerShell

Exemple 1 : Cet exemple enregistre une ligne de base de correctif comme ligne de base de correctif par défaut.

```
Register-SSMDefaultPatchBaseline -BaselineId "pb-03da896ca3b68b639"
```

Sortie :

```
pb-03da896ca3b68b639
```

- Pour API plus de détails, consultez la section RegisterDefaultPatchBaselineRéférence des AWS Tools for PowerShell applets de commande.

### `Register-SSMPatchBaselineForPatchGroup`

L'exemple de code suivant montre comment utiliser`Register-SSMPatchBaselineForPatchGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple enregistre une ligne de base de correctifs pour un groupe de correctifs.

```
Register-SSMPatchBaselineForPatchGroup -BaselineId "pb-03da896ca3b68b639" -
PatchGroup "Production"
```

Sortie :

```
BaselineId           PatchGroup
----------           ----------
pb-03da896ca3b68b639 Production
```

- Pour API plus de détails, consultez la section RegisterPatchBaselineForPatchGroupRéférence des AWS Tools for PowerShell applets de commande.

### `Register-SSMTargetWithMaintenanceWindow`

L'exemple de code suivant montre comment utiliser`Register-SSMTargetWithMaintenanceWindow`.

Outils pour PowerShell

Exemple 1 : Cet exemple enregistre une instance avec une fenêtre de maintenance.

```
$option1 = @{Key="InstanceIds";Values=@("i-0000293ffd8c57862")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

Sortie :

```
d8e47760-23ed-46a5-9f28-927337725398
```

Exemple 2 : Cet exemple enregistre plusieurs instances avec une fenêtre de maintenance.

```
$option1 =
 @{Key="InstanceIds";Values=@("i-0000293ffd8c57862","i-0cb2b964d3e14fd9f")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

Sortie :

```
6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

Exemple 3 : Cet exemple enregistre une instance avec une fenêtre de maintenance à l'aide de EC2 balises.

```
$option1 = @{Key="tag:Environment";Values=@("Production")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Production Web Servers" -ResourceType "INSTANCE"
```

Sortie :

```
2994977e-aefb-4a71-beac-df620352f184
```

- Pour API plus de détails, consultez la section RegisterTargetWithMaintenanceWindowRéférence des AWS Tools for PowerShell applets de commande.

### `Register-SSMTaskWithMaintenanceWindow`

L'exemple de code suivant montre comment utiliser`Register-SSMTaskWithMaintenanceWindow`.

Outils pour PowerShell

Exemple 1 : Cet exemple enregistre une tâche avec une fenêtre de maintenance à l'aide d'un ID d'instance. Le résultat est l'ID de tâche.

```
$parameters = @{}
$parameterValues = New-Object
 Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression
$parameterValues.Values = @("Install")
$parameters.Add("Operation", $parameterValues)

Register-SSMTaskWithMaintenanceWindow -WindowId "mw-03a342e62c96d31b0"
 -ServiceRoleArn "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
 -MaxConcurrency 1 -MaxError 1 -TaskArn "AWS-RunShellScript" -Target
 @{ Key="InstanceIds";Values="i-0000293ffd8c57862" } -TaskType "RUN_COMMAND" -
Priority 10 -TaskParameter $parameters
```

Sortie :

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

Exemple 2 : Cet exemple enregistre une tâche avec une fenêtre de maintenance à l'aide d'un ID cible. Le résultat est l'ID de tâche.

```
$parameters = @{}
$parameterValues = New-Object
 Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression
$parameterValues.Values = @("Install")
$parameters.Add("Operation", $parameterValues)

register-ssmtaskwithmaintenancewindow -WindowId "mw-03a342e62c96d31b0"
 -ServiceRoleArn "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
 -MaxConcurrency 1 -MaxError 1 -TaskArn "AWS-RunShellScript" -Target
 @{ Key="WindowTargetIds";Values="350d44e6-28cc-44e2-951f-4b2c985838f6" } -TaskType
 "RUN_COMMAND" -Priority 10 -TaskParameter $parameters
```

Sortie :

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

Exemple 3 : Cet exemple crée un objet de paramètre pour le document de commande d'exécution **`AWS-RunPowerShellScript`** et crée une tâche avec une fenêtre de maintenance donnée en utilisant l'ID cible. Le résultat renvoyé est l'ID de tâche.

```
$parameters =
 [Collections.Generic.Dictionary[String,Collections.Generic.List[String]]]::new()
$parameters.Add("commands",@("ipconfig","dir env:\computername"))
$parameters.Add("executionTimeout",@(3600))

$props = @{
    WindowId = "mw-0123e4cce56ff78ae"
    ServiceRoleArn = "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
    MaxConcurrency = 1
    MaxError = 1
    TaskType = "RUN_COMMAND"
    TaskArn = "AWS-RunPowerShellScript"
    Target = @{Key="WindowTargetIds";Values="fe1234ea-56d7-890b-12f3-456b789bee0f"}
    Priority = 1
    RunCommand_Parameter = $parameters
    Name = "set-via-cmdlet"
}

Register-SSMTaskWithMaintenanceWindow @props
```

Sortie :

```
f1e2ef34-5678-12e3-456a-12334c5c6cbe
```

Exemple 4 : Cet exemple enregistre une tâche d'automatisation de AWS Systems Manager à l'aide d'un document nommé**`Create-Snapshots`**.

```
$automationParameters = @{}
$automationParameters.Add( "instanceId", @("{{ TARGET_ID }}") )
$automationParameters.Add( "AutomationAssumeRole",
 @("{arn:aws:iam::111111111111:role/AutomationRole}") )
$automationParameters.Add( "SnapshotTimeout", @("PT20M") )
Register-SSMTaskWithMaintenanceWindow -WindowId mw-123EXAMPLE456`
    -ServiceRoleArn "arn:aws:iam::123456789012:role/MW-Role"`
    -MaxConcurrency 1 -MaxError 1 -TaskArn "CreateVolumeSnapshots"`
```

```
    -Target @{ Key="WindowTargetIds";Values="4b5acdf4-946c-4355-
bd68-4329a43a5fd1" }`
    -TaskType "AUTOMATION"`
    -Priority 4`
    -Automation_DocumentVersion '$DEFAULT' -Automation_Parameter
 $automationParameters -Name "Create-Snapshots"
```

- Pour API plus de détails, consultez la section RegisterTaskWithMaintenanceWindowRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-SSMActivation`

L'exemple de code suivant montre comment utiliser`Remove-SSMActivation`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime une activation. Il n'y a aucune sortie si la commande aboutit.

```
Remove-SSMActivation -ActivationId "08e51e79-1e36-446c-8e63-9458569c1363"
```

- Pour API plus de détails, consultez la section DeleteActivationRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-SSMAssociation`

L'exemple de code suivant montre comment utiliser`Remove-SSMAssociation`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime l'association entre une instance et un document. Il n'y a aucune sortie si la commande aboutit.

```
Remove-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

- Pour API plus de détails, consultez la section DeleteAssociationRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-SSMDocument`

L'exemple de code suivant montre comment utiliser`Remove-SSMDocument`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime un document. Il n'y a aucune sortie si la commande aboutit.

```
Remove-SSMDocument -Name "RunShellScript"
```

- Pour API plus de détails, consultez la section DeleteDocumentRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-SSMMaintenanceWindow`

L'exemple de code suivant montre comment utiliser`Remove-SSMMaintenanceWindow`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime une fenêtre de maintenance.

```
Remove-SSMMaintenanceWindow -WindowId "mw-06d59c1a07c022145"
```

Sortie :

```
mw-06d59c1a07c022145
```

- Pour API plus de détails, consultez la section DeleteMaintenanceWindowRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-SSMParameter`

L'exemple de code suivant montre comment utiliser`Remove-SSMParameter`.

Outils pour PowerShell

Exemple 1 : cet exemple supprime un paramètre. Il n'y a aucune sortie si la commande aboutit.

```
Remove-SSMParameter -Name "helloWorld"
```

- Pour API plus de détails, consultez la section DeleteParameterRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-SSMPatchBaseline`

L'exemple de code suivant montre comment utiliser`Remove-SSMPatchBaseline`.

Outils pour PowerShell

Exemple 1 : cet exemple supprime une ligne de base de correctif.

```
Remove-SSMPatchBaseline -BaselineId "pb-045f10b4f382baeda"
```

Sortie :

```
pb-045f10b4f382baeda
```

- Pour API plus de détails, consultez la section DeletePatchBaselineRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-SSMResourceTag`

L'exemple de code suivant montre comment utiliser`Remove-SSMResourceTag`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime une balise d'une fenêtre de maintenance. Il n'y a aucune sortie si la commande aboutit.

```
Remove-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -TagKey "Production"
```

- Pour API plus de détails, consultez la section RemoveTagsFromResourceRéférence des AWS Tools for PowerShell applets de commande.

### `Send-SSMCommand`

L'exemple de code suivant montre comment utiliser`Send-SSMCommand`.

Outils pour PowerShell

Exemple 1 : Cet exemple exécute une commande echo sur une instance cible.

```
Send-SSMCommand -DocumentName "AWS-RunPowerShellScript" -Parameter @{commands =
 "echo helloWorld"} -Target @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
```

Sortie :

```
CommandId          : d8d190fc-32c1-4d65-a0df-ff5ff3965524
Comment            :
CompletedCount     : 0
DocumentName       : AWS-RunPowerShellScript
ErrorCount         : 0
ExpiresAfter       : 3/7/2017 10:48:37 PM
InstanceIds        : {}
MaxConcurrency     : 50
MaxErrors          : 0
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
OutputS3BucketName :
OutputS3KeyPrefix  :
OutputS3Region     :
Parameters         : {[commands,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
RequestedDateTime  : 3/7/2017 9:48:37 PM
ServiceRole        :
Status             : Pending
StatusDetails      : Pending
TargetCount        : 0
Targets            : {instanceids}
```

Exemple 2 : Cet exemple montre comment exécuter une commande qui accepte des paramètres imbriqués.

```
Send-SSMCommand -DocumentName "AWS-RunRemoteScript" -Parameter
 @{ sourceType="GitHub";sourceInfo='{"owner": "me","repository": "amazon-
ssm","path": "Examples/Install-Win32OpenSSH"}'; "commandLine"=".\Install-
Win32OpenSSH.ps1"} -InstanceId i-0cb2b964d3e14fd9f
```

- Pour API plus de détails, consultez la section SendCommandRéférence des AWS Tools for PowerShell applets de commande.

### `Start-SSMAutomationExecution`

L'exemple de code suivant montre comment utiliser`Start-SSMAutomationExecution`.

Outils pour PowerShell

Exemple 1 : Cet exemple exécute un document spécifiant un rôle d'automatisation, un ID AMI source et un rôle d'EC2instance Amazon.

```
Start-SSMAutomationExecution -DocumentName AWS-UpdateLinuxAmi -
Parameter @{'AutomationAssumeRole'='arn:aws:iam::123456789012:role/
SSMAutomationRole';'SourceAmiId'='ami-f173cc91';'InstanceIamRole'='EC2InstanceRole'}
```

Sortie :

```
3a532a4f-0382-11e7-9df7-6f11185f6dd1
```

- Pour API plus de détails, consultez la section StartAutomationExecutionRéférence des AWS Tools for PowerShell applets de commande.

### `Start-SSMSession`

L'exemple de code suivant montre comment utiliser`Start-SSMSession`.

Outils pour PowerShell

Exemple 1 : Cet exemple établit une connexion à une cible pour une session du gestionnaire de session, en activant le transfert de port.

```
Start-SSMSession -Target 'i-064578e5e7454488f' -DocumentName 'AWS-
StartPortForwardingSession' -Parameter @{ localPortNumber = '8080'; portNumber =
 '80' }
```

Sortie :

```
SessionId    StreamUrl
----------    ----------
random-id0    wss://ssmmessages.amazonaws.com/v1/data-channel/random-id
```

- Pour API plus de détails, consultez la section StartSessionRéférence des AWS Tools for PowerShell applets de commande.

### `Stop-SSMAutomationExecution`

L'exemple de code suivant montre comment utiliser`Stop-SSMAutomationExecution`.

Outils pour PowerShell

Exemple 1 : Cet exemple arrête une exécution automatique. Il n'y a aucune sortie si la commande aboutit.

```
Stop-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"
```

- Pour API plus de détails, consultez la section StopAutomationExecutionRéférence des AWS Tools for PowerShell applets de commande.

### `Stop-SSMCommand`

L'exemple de code suivant montre comment utiliser`Stop-SSMCommand`.

Outils pour PowerShell

Exemple 1 : Cet exemple tente d'annuler une commande. Il n'y a aucune sortie si l'opération réussit.

```
Stop-SSMCommand -CommandId "9ded293e-e792-4440-8e3e-7b8ec5feaa38"
```

- Pour API plus de détails, consultez la section CancelCommandRéférence des AWS Tools for PowerShell applets de commande.

### `Unregister-SSMManagedInstance`

L'exemple de code suivant montre comment utiliser`Unregister-SSMManagedInstance`.

Outils pour PowerShell

Exemple 1 : Cet exemple annule l'enregistrement d'une instance gérée. Il n'y a aucune sortie si la commande aboutit.

```
Unregister-SSMManagedInstance -InstanceId "mi-08ab247cdf1046573"
```

- Pour API plus de détails, consultez la section DeregisterManagedInstanceRéférence des AWS Tools for PowerShell applets de commande.

### `Unregister-SSMPatchBaselineForPatchGroup`

L'exemple de code suivant montre comment utiliser`Unregister-SSMPatchBaselineForPatchGroup`.

Outils pour PowerShell

Exemple 1 : cet exemple désenregistre un groupe de correctifs d'une ligne de base de correctifs.

```
Unregister-SSMPatchBaselineForPatchGroup -BaselineId "pb-045f10b4f382baeda" -
PatchGroup "Production"
```

Sortie :

```
BaselineId           PatchGroup
----------           ----------
pb-045f10b4f382baeda Production
```

- Pour API plus de détails, consultez la section DeregisterPatchBaselineForPatchGroupRéférence des AWS Tools for PowerShell applets de commande.

### `Unregister-SSMTargetFromMaintenanceWindow`

L'exemple de code suivant montre comment utiliser`Unregister-SSMTargetFromMaintenanceWindow`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime une cible d'une fenêtre de maintenance.

```
Unregister-SSMTargetFromMaintenanceWindow -WindowTargetId "6ab5c208-9fc4-4697-84b7-
b02a6cc25f7d" -WindowId "mw-06cf17cbefcb4bf4f"
```

Sortie :

```
WindowId             WindowTargetId
```

```
--------             --------------
mw-06cf17cbefcb4bf4f 6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

- Pour API plus de détails, consultez la section DeregisterTargetFromMaintenanceWindowRéférence des AWS Tools for PowerShell applets de commande.

### `Unregister-SSMTaskFromMaintenanceWindow`

L'exemple de code suivant montre comment utiliser`Unregister-SSMTaskFromMaintenanceWindow`.

Outils pour PowerShell

Exemple 1 : Cet exemple supprime une tâche d'une fenêtre de maintenance.

```
Unregister-SSMTaskFromMaintenanceWindow -WindowTaskId "f34a2c47-ddfd-4c85-
a88d-72366b69af1b" -WindowId "mw-03a342e62c96d31b0"
```

Sortie :

```
WindowId             WindowTaskId
--------             ------------
mw-03a342e62c96d31b0 f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

- Pour API plus de détails, consultez la section DeregisterTaskFromMaintenanceWindowRéférence des AWS Tools for PowerShell applets de commande.

### `Update-SSMAssociation`

L'exemple de code suivant montre comment utiliser`Update-SSMAssociation`.

Outils pour PowerShell

Exemple 1 : Cet exemple met à jour une association avec une nouvelle version de document.

```
Update-SSMAssociation -AssociationId "93285663-92df-44cb-9f26-2292d4ecc439" -
DocumentVersion "1"
```

Sortie :

```
Name                  : AWS-UpdateSSMAgent
InstanceId            :
Date                  : 3/1/2017 6:22:21 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

- Pour API plus de détails, consultez la section UpdateAssociationRéférence des AWS Tools for PowerShell applets de commande.

### `Update-SSMAssociationStatus`

L'exemple de code suivant montre comment utiliser`Update-SSMAssociationStatus`.

Outils pour PowerShell

Exemple 1 : Cet exemple met à jour le statut de l'association entre une instance et un document de configuration.

```
Update-SSMAssociationStatus -Name "AWS-UpdateSSMAgent" -InstanceId
 "i-0000293ffd8c57862" -AssociationStatus_Date "2015-02-20T08:31:11Z"
 -AssociationStatus_Name "Pending" -AssociationStatus_Message
 "temporary_status_change" -AssociationStatus_AdditionalInfo "Additional-Config-
Needed"
```

Sortie :

```
Name                  : AWS-UpdateSSMAgent
InstanceId            : i-0000293ffd8c57862
Date                  : 2/23/2017 6:55:22 PM
Status.Name           : Pending
Status.Date           : 2/20/2015 8:31:11 AM
Status.Message        : temporary_status_change
Status.AdditionalInfo : Additional-Config-Needed
```

- Pour API plus de détails, consultez la section UpdateAssociationStatusRéférence des AWS Tools for PowerShell applets de commande.

### `Update-SSMDocument`

L'exemple de code suivant montre comment utiliser`Update-SSMDocument`.

Outils pour PowerShell

Exemple 1 : Cela crée une nouvelle version d'un document avec le contenu mis à jour du fichier JSON que vous spécifiez. Le document doit être au JSON format. Vous pouvez obtenir la version du document à l'aide de l'applet de commande « Get- SSMDocumentVersionList ».

```
Update-SSMDocument -Name RunShellScript -DocumentVersion "1" -Content (Get-Content -
Raw "c:\temp\RunShellScript.json")
```

Sortie :

```
CreatedDate     : 3/1/2017 2:59:17 AM
DefaultVersion  : 1
Description     : Run an updated script
DocumentType    : Command
DocumentVersion : 2
Hash            : 1d5ce820e999ff051eb4841ed887593daf77120fd76cae0d18a53cc42e4e22c1
HashType        : Sha256
LatestVersion   : 2
Name            : RunShellScript
Owner           : 809632081692
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion   : 2.0
Sha1            :
Status          : Updating
```

- Pour API plus de détails, consultez la section UpdateDocumentRéférence des AWS Tools for PowerShell applets de commande.

### `Update-SSMDocumentDefaultVersion`

L'exemple de code suivant montre comment utiliser`Update-SSMDocumentDefaultVersion`.

Outils pour PowerShell

Exemple 1 : Ceci met à jour la version par défaut d'un document. Vous pouvez obtenir les versions de document disponibles à l'aide de l'applet de commande « Get-SSMDocumentVersionList ».

```
Update-SSMDocumentDefaultVersion -Name "RunShellScript" -DocumentVersion "2"
```

Sortie :

```
DefaultVersion Name
-------------- ----
2              RunShellScript
```

- Pour API plus de détails, consultez la section UpdateDocumentDefaultVersionRéférence des AWS Tools for PowerShell applets de commande.

### `Update-SSMMaintenanceWindow`

L'exemple de code suivant montre comment utiliser`Update-SSMMaintenanceWindow`.

Outils pour PowerShell

Exemple 1 : Cet exemple met à jour le nom d'une fenêtre de maintenance.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Name "My-Renamed-MW"
```

Sortie :

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

Exemple 2 : Cet exemple active une fenêtre de maintenance.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $true
```

Sortie :

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

Exemple 3 : Cet exemple désactive une fenêtre de maintenance.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $false
```

Sortie :

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : False
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

- Pour API plus de détails, consultez la section UpdateMaintenanceWindowRéférence des AWS Tools for PowerShell applets de commande.

### `Update-SSMManagedInstanceRole`

L'exemple de code suivant montre comment utiliser`Update-SSMManagedInstanceRole`.

Outils pour PowerShell

Exemple 1 : Cet exemple met à jour le rôle d'une instance gérée. Il n'y a aucune sortie si la commande aboutit.

```
Update-SSMManagedInstanceRole -InstanceId "mi-08ab247cdf1046573" -IamRole
 "AutomationRole"
```

- Pour API plus de détails, consultez la section UpdateManagedInstanceRoleRéférence des AWS Tools for PowerShell applets de commande.

### `Update-SSMPatchBaseline`

L'exemple de code suivant montre comment utiliser`Update-SSMPatchBaseline`.

Outils pour PowerShell

Exemple 1 : Cet exemple ajoute deux correctifs rejetés et un correctif approuvé à une ligne de base de correctifs existante.

```
Update-SSMPatchBaseline -BaselineId "pb-03da896ca3b68b639" -RejectedPatch
 "KB2032276","MS10-048" -ApprovedPatch "KB2124261"
```

Sortie :

```
ApprovalRules   : Amazon.SimpleSystemsManagement.Model.PatchRuleGroup
ApprovedPatches : {KB2124261}
BaselineId      : pb-03da896ca3b68b639
CreatedDate     : 3/3/2017 5:02:19 PM
Description     : Baseline containing all updates approved for production systems
GlobalFilters   : Amazon.SimpleSystemsManagement.Model.PatchFilterGroup
ModifiedDate    : 3/3/2017 5:22:10 PM
Name            : Production-Baseline
RejectedPatches : {KB2032276, MS10-048}
```

- Pour API plus de détails, consultez la section UpdatePatchBaselineRéférence des AWS Tools for PowerShell applets de commande.

### `Write-SSMComplianceItem`

L'exemple de code suivant montre comment utiliser`Write-SSMComplianceItem`.

Outils pour PowerShell

Exemple 1 : Cet exemple écrit un élément de conformité personnalisé pour l'instance gérée donnée

```
$item = [Amazon.SimpleSystemsManagement.Model.ComplianceItemEntry]::new()
$item.Id = "07Jun2019-3"
$item.Severity="LOW"
$item.Status="COMPLIANT"
$item.Title="Fin-test-1 - custom"
```

```
Write-SSMComplianceItem -ResourceId mi-012dcb3ecea45b678 -ComplianceType
 Custom:VSSCompliant2 -ResourceType ManagedInstance -Item $item -
ExecutionSummary_ExecutionTime "07-Jun-2019"
```

- Pour API plus de détails, consultez la section PutComplianceItemsRéférence des AWS Tools for PowerShell applets de commande.

### `Write-SSMInventory`

L'exemple de code suivant montre comment utiliser`Write-SSMInventory`.

Outils pour PowerShell

Exemple 1 : Cet exemple attribue des informations sur l'emplacement du rack à une instance. Il n'y a aucune sortie si la commande aboutit.

```
$data = New-Object
 "System.Collections.Generic.Dictionary[System.String,System.String]"
$data.Add("RackLocation", "Bay B/Row C/Rack D/Shelf F")

$items = New-Object
 "System.Collections.Generic.List[System.Collections.Generic.Dictionary[System.String,
 System.String]]"
$items.Add($data)

$customInventoryItem = New-Object Amazon.SimpleSystemsManagement.Model.InventoryItem
$customInventoryItem.CaptureTime = "2016-08-22T10:01:01Z"
$customInventoryItem.Content = $items
$customInventoryItem.TypeName = "Custom:TestRackInfo2"
$customInventoryItem.SchemaVersion = "1.0"

$inventoryItems = @($customInventoryItem)

Write-SSMInventory -InstanceId "i-0cb2b964d3e14fd9f" -Item $inventoryItems
```

- Pour API plus de détails, consultez la section PutInventoryRéférence des AWS Tools for PowerShell applets de commande.

### `Write-SSMParameter`

L'exemple de code suivant montre comment utiliser`Write-SSMParameter`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un paramètre. Il n'y a aucune sortie si la commande aboutit.

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "helloWorld"
```

Exemple 2 : Cet exemple modifie un paramètre. Il n'y a aucune sortie si la commande aboutit.

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "Good day, Sunshine!" -
Overwrite $true
```

- Pour API plus de détails, consultez la section PutParameterRéférence des AWS Tools for PowerShell applets de commande.

# Exemples d'Amazon Translate utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS Tools for PowerShell aide d'Amazon Translate.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `ConvertTo-TRNTargetLanguage`

L'exemple de code suivant montre comment utiliser`ConvertTo-TRNTargetLanguage`.

Outils pour PowerShell

Exemple 1 : convertit le texte anglais spécifié en français. Le texte à convertir peut également être transmis en tant que paramètre -Text.

```
"Hello World" | ConvertTo-TRNTargetLanguage -SourceLanguageCode en -
TargetLanguageCode fr
```

- Pour API plus de détails, consultez la section TranslateTextRéférence des AWS Tools for PowerShell applets de commande.

# AWS WAFV2 exemples utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS Tools for PowerShell with AWS WAFV2.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `New-WAF2WebACL`

L'exemple de code suivant montre comment utiliser`New-WAF2WebACL`.

Outils pour PowerShell

Exemple 1 : Cette commande crée un nouveau site Web ACL nommé « waf-test ». Veuillez noter que selon API la documentation du service, « DefaultAction » est une propriété obligatoire. Par conséquent, la valeur de «- DefaultAction _Allow » et/ou de «- DefaultAction _Block » doit être spécifiée. Étant donné que «- DefaultAction _Allow » et «- DefaultAction _Block » ne sont pas les propriétés requises, la valeur « @ {} » peut être utilisée comme espace réservé, comme indiqué dans l'exemple ci-dessus.

```
New-WAF2WebACL -Name "waf-test" -Scope REGIONAL -Region eu-
west-1 -VisibilityConfig_CloudWatchMetricsEnabled $true -
```

```
VisibilityConfig_SampledRequestsEnabled $true -VisibilityConfig_MetricName "waf-
test" -Description "Test" -DefaultAction_Allow @{}
```

Sortie :

```
ARN         : arn:aws:wafv2:eu-west-1:139480602983:regional/webacl/waf-
test/19460b3f-db14-4b9a-8e23-a417e1eb007f
Description : Test
Id          : 19460b3f-db14-4b9a-8e23-a417e1eb007f
LockToken   : 5a0cd5eb-d911-4341-b313-b429e6d6b6ab
Name        : waf-test
```

- Pour API plus de détails, consultez la section CreateWebAclRéférence des AWS Tools for PowerShell applets de commande.

# WorkSpaces exemples utilisant des outils pour PowerShell

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS Tools for PowerShell with WorkSpaces.

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

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- Actions

## Actions

### `Approve-WKSIpRule`

L'exemple de code suivant montre comment utiliser`Approve-WKSIpRule`.

Outils pour PowerShell

Exemple 1 : cet exemple ajoute des règles à un groupe IP existant

```
$Rule = @(
@{IPRule = "10.1.0.0/0"; RuleDesc = "First Rule Added"},
@{IPRule = "10.2.0.0/0"; RuleDesc = "Second Rule Added"}
)

Approve-WKSIpRule -GroupId wsipg-abcnx2fcw -UserRule $Rule
```

- Pour API plus de détails, consultez la section AuthorizeIpRulesRéférence des AWS Tools for PowerShell applets de commande.

### `Copy-WKSWorkspaceImage`

L'exemple de code suivant montre comment utiliser`Copy-WKSWorkspaceImage`.

Outils pour PowerShell

Exemple 1 : Cet exemple copie l'image de l'espace de travail avec l'ID spécifié de us-west-2 vers la région actuelle avec le nom « » CopiedImageTest

```
Copy-WKSWorkspaceImage -Name CopiedImageTest -SourceRegion us-west-2 -SourceImageId
 wsi-djfoedhw6
```

Sortie :

```
wsi-456abaqfe
```

- Pour API plus de détails, consultez la section CopyWorkspaceImageRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-WKSClientProperty`

L'exemple de code suivant montre comment utiliser`Edit-WKSClientProperty`.

Outils pour PowerShell

Exemple 1 : Cet exemple active la reconnexion pour le client Workspaces

```
Edit-WKSClientProperty -Region us-west-2 -ClientProperties_ReconnectEnabled
 "ENABLED" -ResourceId d-123414a369
```

- Pour API plus de détails, consultez la section ModifyClientPropertiesRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-WKSSelfServicePermission`

L'exemple de code suivant montre comment utiliser`Edit-WKSSelfServicePermission`.

Outils pour PowerShell

Exemple 1 : Cet exemple active les autorisations en libre-service pour modifier le type de calcul et augmenter la taille du volume pour le répertoire spécifié

```
Edit-WKSSelfservicePermission -Region us-west-2 -ResourceId
 d-123454a369 -SelfservicePermissions_ChangeComputeType ENABLED -
SelfservicePermissions_IncreaseVolumeSize ENABLED
```

- Pour API plus de détails, consultez la section ModifySelfservicePermissionsRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-WKSWorkspaceAccessProperty`

L'exemple de code suivant montre comment utiliser`Edit-WKSWorkspaceAccessProperty`.

Outils pour PowerShell

Exemple 1 : Cet exemple active l'accès à Workspace sur Android et Chrome OS pour le répertoire spécifié

```
Edit-WKSWorkspaceAccessProperty -Region us-west-2 -ResourceId
 d-123454a369 -WorkspaceAccessProperties_DeviceTypeAndroid ALLOW -
WorkspaceAccessProperties_DeviceTypeChromeOs ALLOW
```

- Pour API plus de détails, consultez la section ModifyWorkspaceAccessPropertiesRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-WKSWorkspaceCreationProperty`

L'exemple de code suivant montre comment utiliser`Edit-WKSWorkspaceCreationProperty`.

Outils pour PowerShell

Exemple 1 : Cet exemple active l'accès à Internet et le mode de maintenance sur true comme valeurs par défaut lors de la création d'un espace de travail

```
Edit-WKSWorkspaceCreationProperty -Region us-west-2 -ResourceId
 d-123454a369 -WorkspaceCreationProperties_EnableInternetAccess $true -
WorkspaceCreationProperties_EnableMaintenanceMode $true
```

- Pour API plus de détails, consultez la section ModifyWorkspaceCreationPropertiesRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-WKSWorkspaceProperty`

L'exemple de code suivant montre comment utiliser`Edit-WKSWorkspaceProperty`.

Outils pour PowerShell

Exemple 1 : Cet exemple modifie la propriété Workspace Running Mode en Auto Stop pour l'espace de travail spécifié

```
Edit-WKSWorkspaceProperty -WorkspaceId ws-w361s100v -Region us-west-2 -
WorkspaceProperties_RunningMode AUTO_STOP
```

- Pour API plus de détails, consultez la section ModifyWorkspacePropertiesRéférence des AWS Tools for PowerShell applets de commande.

### `Edit-WKSWorkspaceState`

L'exemple de code suivant montre comment utiliser`Edit-WKSWorkspaceState`.

Outils pour PowerShell

Exemple 1 : Cet exemple fait passer l'état de l'espace de travail spécifié à Disponible

```
Edit-WKSWorkspaceState -WorkspaceId ws-w361s100v -Region us-west-2 -WorkspaceState
 AVAILABLE
```

- Pour API plus de détails, consultez la section ModifyWorkspaceStateRéférence des AWS Tools for PowerShell applets de commande.

### `Get-WKSClientProperty`

L'exemple de code suivant montre comment utiliser`Get-WKSClientProperty`.

Outils pour PowerShell

Exemple 1 : Cet exemple obtient les propriétés client du client Workspace pour le répertoire spécifié

```
Get-WKSClientProperty -ResourceId d-223562a123
```

- Pour API plus de détails, consultez la section DescribeClientPropertiesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-WKSIpGroup`

L'exemple de code suivant montre comment utiliser`Get-WKSIpGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple obtient les détails du groupe IP spécifié dans la région spécifiée

```
Get-WKSIpGroup -Region us-east-1 -GroupId wsipg-8m1234v45
```

Sortie :

```
GroupDesc GroupId         GroupName UserRules
--------- -------         --------- ---------
          wsipg-8m1234v45 TestGroup {Amazon.WorkSpaces.Model.IpRuleItem,
 Amazon.WorkSpaces.Model.IpRuleItem}
```

- Pour API plus de détails, consultez la section DescribeIpGroupsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-WKSTag`

L'exemple de code suivant montre comment utiliser`Get-WKSTag`.

Outils pour PowerShell

Exemple 1 : cet exemple récupère la balise pour l'espace de travail donné

```
Get-WKSTag -WorkspaceId ws-w361s234r -Region us-west-2
```

Sortie :

```
Key         Value
---         -----
auto-delete no
purpose     Workbench
```

- Pour API plus de détails, consultez la section DescribeTagsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-WKSWorkspace`

L'exemple de code suivant montre comment utiliser`Get-WKSWorkspace`.

Outils pour PowerShell

Exemple 1 : récupère les détails de toutes vos connexions WorkSpaces au pipeline.

```
Get-WKSWorkspace
```

Sortie :

```
BundleId                   : wsb-1a2b3c4d
ComputerName               :
DirectoryId                : d-1a2b3c4d
ErrorCode                  :
ErrorMessage               :
IpAddress                  :
RootVolumeEncryptionEnabled : False
State                      : PENDING
SubnetId                   :
UserName                   : myuser
UserVolumeEncryptionEnabled : False
VolumeEncryptionKey        :
WorkspaceId                : ws-1a2b3c4d
WorkspaceProperties        : Amazon.WorkSpaces.Model.WorkspaceProperties
```

Exemple 2 : Cette commande affiche les valeurs des propriétés enfant **`WorkSpaceProperties`** d'un espace de travail de la **`us-west-2`** région. Pour plus d'informations sur les propriétés

enfant de**`WorkSpaceProperties`**, consultez https://docs.aws.amazon.com/workspaces/ API_WorkspaceProperties latest/api/ .html.

```
(Get-WKSWorkspace -Region us-west-2 -WorkSpaceId ws-xdaf7hc9s).WorkspaceProperties
```

Sortie :

```
        ComputeTypeName                     : STANDARD
        RootVolumeSizeGib                   : 80
        RunningMode                         : AUTO_STOP
        RunningModeAutoStopTimeoutInMinutes : 60
        UserVolumeSizeGib                   : 50
```

Exemple 3 : Cette commande indique la valeur de la propriété enfant **`RootVolumeSizeGib`** de **`WorkSpaceProperties`** pour un espace de travail de la **`us-west-2`** région. La taille du volume racine, en GiB, est de 80.

```
(Get-WKSWorkspace -Region us-west-2 -WorkSpaceId ws-
xdaf7hc9s).WorkspaceProperties.RootVolumeSizeGib
```

Sortie :

```
        80
```

- Pour API plus de détails, consultez la section DescribeWorkspacesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-WKSWorkspaceBundle`

L'exemple de code suivant montre comment utiliser`Get-WKSWorkspaceBundle`.

Outils pour PowerShell

Exemple 1 : cet exemple récupère les détails de tous les packs Workspace de la région actuelle

```
Get-WKSWorkspaceBundle
```

Sortie :

```
BundleId        : wsb-sfhdgv342
ComputeType     : Amazon.WorkSpaces.Model.ComputeType
Description     : This bundle is custom
ImageId         : wsi-235aeqges
LastUpdatedTime : 12/26/2019 06:44:07
Name            : CustomBundleTest
Owner           : 233816212345
RootStorage     : Amazon.WorkSpaces.Model.RootStorage
UserStorage     : Amazon.WorkSpaces.Model.UserStorage
```

- Pour API plus de détails, consultez la section DescribeWorkspaceBundlesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-WKSWorkspaceDirectory`

L'exemple de code suivant montre comment utiliser`Get-WKSWorkspaceDirectory`.

Outils pour PowerShell

Exemple 1 : Cet exemple répertorie les détails des annuaires enregistrés

```
Get-WKSWorkspaceDirectory
```

Sortie :

```
Alias                       : TestWorkspace
CustomerUserName            : Administrator
DirectoryId                 : d-123414a369
DirectoryName               : TestDirectory.com
DirectoryType               : MicrosoftAD
DnsIpAddresses              : {172.31.43.45, 172.31.2.97}
IamRoleId                   : arn:aws:iam::761234567801:role/workspaces_RoleDefault
IpGroupIds                  : {}
RegistrationCode            : WSpdx+4RRT43
SelfservicePermissions      : Amazon.WorkSpaces.Model.SelfservicePermissions
State                       : REGISTERED
SubnetIds                   : {subnet-1m3m7b43, subnet-ard11aba}
Tenancy                     : SHARED
WorkspaceAccessProperties   : Amazon.WorkSpaces.Model.WorkspaceAccessProperties
WorkspaceCreationProperties :
 Amazon.WorkSpaces.Model.DefaultWorkspaceCreationProperties
```

```
WorkspaceSecurityGroupId    : sg-0ed2441234a123c43
```

- Pour API plus de détails, consultez la section DescribeWorkspaceDirectoriesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-WKSWorkspaceImage`

L'exemple de code suivant montre comment utiliser`Get-WKSWorkspaceImage`.

Outils pour PowerShell

Exemple 1 : cet exemple récupère tous les détails de toutes les images de la région

```
Get-WKSWorkspaceImage
```

Sortie :

```
Description     :This image is copied from another image
ErrorCode       :
ErrorMessage    :
ImageId         : wsi-345ahdjgo
Name            : CopiedImageTest
OperatingSystem : Amazon.WorkSpaces.Model.OperatingSystem
RequiredTenancy : DEFAULT
State           : AVAILABLE
```

- Pour API plus de détails, consultez la section DescribeWorkspaceImagesRéférence des AWS Tools for PowerShell applets de commande.

### `Get-WKSWorkspaceSnapshot`

L'exemple de code suivant montre comment utiliser`Get-WKSWorkspaceSnapshot`.

Outils pour PowerShell

Exemple 1 : Cet exemple montre l'horodatage du dernier instantané créé pour l'espace de travail spécifié

```
Get-WKSWorkspaceSnapshot -WorkspaceId ws-w361s100v
```

Sortie :

```
RebuildSnapshots                  RestoreSnapshots
----------------                  ----------------
{Amazon.WorkSpaces.Model.Snapshot} {Amazon.WorkSpaces.Model.Snapshot}
```

- Pour API plus de détails, consultez la section DescribeWorkspaceSnapshotsRéférence des AWS Tools for PowerShell applets de commande.

### `Get-WKSWorkspacesConnectionStatus`

L'exemple de code suivant montre comment utiliser`Get-WKSWorkspacesConnectionStatus`.

Outils pour PowerShell

Exemple 1 : Cet exemple extrait l'état de connexion pour l'espace de travail spécifié

```
Get-WKSWorkspacesConnectionStatus -WorkspaceId ws-w123s234r
```

- Pour API plus de détails, consultez la section DescribeWorkspacesConnectionStatusRéférence des AWS Tools for PowerShell applets de commande.

### `New-WKSIpGroup`

L'exemple de code suivant montre comment utiliser`New-WKSIpGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple crée un groupe d'adresses IP vide nommé FreshEmptyIpGroup

```
New-WKSIpGroup -GroupName "FreshNewIPGroup"
```

Sortie :

```
wsipg-w45rty4ty
```

- Pour API plus de détails, consultez la section CreateIpGroupRéférence des AWS Tools for PowerShell applets de commande.

### `New-WKSTag`

L'exemple de code suivant montre comment utiliser`New-WKSTag`.

Outils pour PowerShell

Exemple 1 : Cet exemple ajoute une nouvelle balise à un espace de travail nommé**`ws-wsname`**. La balise possède une clé de « Nom » et une valeur de clé de**`AWS_Workspace`**.

```
$tag = New-Object Amazon.WorkSpaces.Model.Tag
$tag.Key = "Name"
$tag.Value = "AWS_Workspace"
New-WKSTag -Region us-west-2 -WorkspaceId ws-wsname -Tag $tag
```

Exemple 2 : Cet exemple ajoute plusieurs balises à un espace de travail nommé**`ws-wsname`**. Une balise possède une clé « Nom » et une valeur clé de **`AWS_Workspace`** ; l'autre balise a une clé « Stage » et une valeur clé « Test ».

```
$tag = New-Object Amazon.WorkSpaces.Model.Tag
$tag.Key = "Name"
$tag.Value = "AWS_Workspace"

$tag2 = New-Object Amazon.WorkSpaces.Model.Tag
$tag2.Key = "Stage"
$tag2.Value = "Test"
New-WKSTag -Region us-west-2 -WorkspaceId ws-wsname -Tag $tag,$tag2
```

- Pour API plus de détails, consultez la section CreateTagsRéférence des AWS Tools for PowerShell applets de commande.

### `New-WKSWorkspace`

L'exemple de code suivant montre comment utiliser`New-WKSWorkspace`.

Outils pour PowerShell

Exemple 1 : créez un WorkSpace pour le bundle, le répertoire et l'utilisateur fournis.

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId" =
 "d-1a2b3c4d"; "UserName" = "USERNAME"}
```

Exemple 2 : Cet exemple crée plusieurs WorkSpaces

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId"
 = "d-1a2b3c4d"; "UserName" = "USERNAME_1"},@{"BundleID" = "wsb-1a2b3c4d";
 "DirectoryId" = "d-1a2b3c4d"; "UserName" = "USERNAME_2"}
```

- Pour API plus de détails, consultez la section CreateWorkspacesRéférence des AWS Tools for PowerShell applets de commande.

### `Register-WKSIpGroup`

L'exemple de code suivant montre comment utiliser`Register-WKSIpGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple enregistre le groupe IP spécifié dans le répertoire spécifié

```
Register-WKSIpGroup -GroupId wsipg-23ahsdres -DirectoryId d-123412e123
```

- Pour API plus de détails, consultez la section AssociateIpGroupsRéférence des AWS Tools for PowerShell applets de commande.

### `Register-WKSWorkspaceDirectory`

L'exemple de code suivant montre comment utiliser`Register-WKSWorkspaceDirectory`.

Outils pour PowerShell

Exemple 1 : Cet exemple enregistre le répertoire spécifié pour le service Workspaces

```
Register-WKSWorkspaceDirectory -DirectoryId d-123412a123 -EnableWorkDoc $false
```

- Pour API plus de détails, consultez la section RegisterWorkspaceDirectoryRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-WKSIpGroup`

L'exemple de code suivant montre comment utiliser`Remove-WKSIpGroup`.

Outils pour PowerShell

Exemple 1 : cet exemple supprime le groupe IP spécifié

```
Remove-WKSIpGroup -GroupId wsipg-32fhgtred
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-WKSIpGroup (DeleteIpGroup)" on target
 "wsipg-32fhgtred".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Pour API plus de détails, consultez la section DeleteIpGroupRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-WKSTag`

L'exemple de code suivant montre comment utiliser`Remove-WKSTag`.

Outils pour PowerShell

Exemple 1 : cet exemple supprime la balise associée à l'espace de travail

```
Remove-WKSTag -ResourceId ws-w10b3abcd -TagKey "Type"
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-WKSTag (DeleteTags)" on target "ws-w10b3abcd".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Pour API plus de détails, consultez la section DeleteTagsRéférence des AWS Tools for PowerShell applets de commande.

### `Remove-WKSWorkspace`

L'exemple de code suivant montre comment utiliser`Remove-WKSWorkspace`.

Outils pour PowerShell

Exemple 1 : met fin à plusieurs WorkSpaces. L'utilisation du commutateur -Force empêche l'applet de commande de demander une confirmation.

```
Remove-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0" -Force
```

Exemple 2 : Récupère la collection de tous vos WorkSpaces et les dirige IDs vers le WorkSpaceId paramètre - de Remove-WKSWorkspace, en mettant fin à tous les. WorkSpaces L'applet de commande affiche un message avant de terminer chacune d'entre elles WorkSpace . Pour supprimer l'invite de confirmation, ajoutez le commutateur -Force.

```
Get-WKSWorkspaces | Remove-WKSWorkspace
```

Exemple 3 : Cet exemple montre comment transmettre TerminateRequest des objets définissant le WorkSpaces à terminer. L'applet de commande demandera une confirmation avant de continuer, sauf si le paramètre de commutation -Force est également spécifié.

```
$arrRequest = @()
$request1 = New-Object Amazon.WorkSpaces.Model.TerminateRequest
$request1.WorkspaceId = 'ws-12345678'
$arrRequest += $request1
$request2 = New-Object Amazon.WorkSpaces.Model.TerminateRequest
$request2.WorkspaceId = 'ws-abcdefgh'
$arrRequest += $request2
Remove-WKSWorkspace -Request $arrRequest
```

- Pour API plus de détails, consultez la section TerminateWorkspacesRéférence des AWS Tools for PowerShell applets de commande.

### `Reset-WKSWorkspace`

L'exemple de code suivant montre comment utiliser`Reset-WKSWorkspace`.

Outils pour PowerShell

Exemple 1 : reconstruit le fichier spécifié WorkSpace.

```
Reset-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

Exemple 2 : récupère la collection de tous vos fichiers WorkSpaces et les redirige IDs vers le WorkSpaceId paramètre - de Reset-WKSWorkspace, ce qui entraîne WorkSpaces leur reconstruction.

```
Get-WKSWorkspaces | Reset-WKSWorkspace
```

- Pour API plus de détails, consultez la section RebuildWorkspacesRéférence des AWS Tools for PowerShell applets de commande.

### `Restart-WKSWorkspace`

L'exemple de code suivant montre comment utiliser`Restart-WKSWorkspace`.

Outils pour PowerShell

Exemple 1 : redémarre le paramètre spécifié WorkSpace.

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

Exemple 2 : redémarre plusieurs WorkSpaces.

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d","ws-5a6b7c8d"
```

Exemple 3 : récupère la collection de tous vos fichiers WorkSpaces et les redirige IDs vers le WorkSpaceId paramètre - de Restart-WKSWorkspace, provoquant le WorkSpaces redémarrage du.

```
Get-WKSWorkspaces | Restart-WKSWorkspace
```

- Pour API plus de détails, consultez la section RebootWorkspacesRéférence des AWS Tools for PowerShell applets de commande.

### `Stop-WKSWorkspace`

L'exemple de code suivant montre comment utiliser`Stop-WKSWorkspace`.

Outils pour PowerShell

Exemple 1 : arrête plusieurs WorkSpaces.

```
Stop-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0"
```

Exemple 2 : récupère la collection de tous vos WorkSpaces fichiers et les dirige IDs vers le WorkSpaceId paramètre - de Stop- WKSWorkspace provoquant WorkSpaces leur arrêt.

```
Get-WKSWorkspaces | Stop-WKSWorkspace
```

Exemple 3 : Cet exemple montre comment transmettre StopRequest des objets définissant le WorkSpaces à arrêter.

```
$arrRequest = @()
$request1 = New-Object Amazon.WorkSpaces.Model.StopRequest
$request1.WorkspaceId = 'ws-12345678'
$arrRequest += $request1
$request2 = New-Object Amazon.WorkSpaces.Model.StopRequest
$request2.WorkspaceId = 'ws-abcdefgh'
$arrRequest += $request2
Stop-WKSWorkspace -Request $arrRequest
```

- Pour API plus de détails, consultez la section StopWorkspacesRéférence des AWS Tools for PowerShell applets de commande.

### `Unregister-WKSIpGroup`

L'exemple de code suivant montre comment utiliser`Unregister-WKSIpGroup`.

Outils pour PowerShell

Exemple 1 : Cet exemple annule l'enregistrement du groupe IP spécifié dans le répertoire spécifié

```
Unregister-WKSIpGroup -GroupId wsipg-12abcdphq -DirectoryId d-123454b123
```

- Pour API plus de détails, consultez la section DisassociateIpGroupsRéférence des AWS Tools for PowerShell applets de commande.

# Sécurité de ce AWS produit ou service

Chez Amazon Web Services (AWS), la sécurité dans le cloud est la priorité principale. En tant que client AWS , vous bénéficiez d'un centre de données et d'une architecture réseau conçus pour répondre aux exigences des organisations les plus pointilleuses sur la sécurité. La sécurité est une responsabilité partagée entre vous AWS et vous. Le modèle de responsabilité partagée décrit cela comme la sécurité du cloud et la sécurité dans le cloud.

Sécurité du cloud : AWS est chargée de protéger l'infrastructure qui exécute tous les services proposés dans le AWS cloud et de vous fournir des services que vous pouvez utiliser en toute sécurité. Notre responsabilité en matière de sécurité est notre priorité absolue AWS, et l'efficacité de notre sécurité est régulièrement testée et vérifiée par des auditeurs tiers dans le cadre des programmes de AWS conformité.

Sécurité dans le cloud — Votre responsabilité est déterminée par le AWS service que vous utilisez et par d'autres facteurs, notamment la sensibilité de vos données, les exigences de votre organisation et les lois et réglementations applicables.

Ce AWS produit ou service suit le modèle de responsabilité partagée par le biais des services Amazon Web Services (AWS) spécifiques qu'il prend en charge. Pour obtenir des informations sur la sécurité des AWS services, consultez la AWS page de documentation sur la sécuritéAWS des services et les services concernés par les efforts de AWS conformité par programme de conformité.

Rubriques



## Protection des données dans ce AWS produit ou service

Le modèle de responsabilité AWS partagée de s'applique à la protection des données dans ce AWS produit ou service. Comme décrit dans ce modèle, AWS est chargé de protéger l'infrastructure mondiale qui gère tous les AWS Cloud. La gestion du contrôle de votre contenu hébergé sur

cette infrastructure relève de votre responsabilité. Vous êtes également responsable des tâches de configuration et de gestion de la sécurité des Services AWS que vous utilisez. Pour plus d'informations sur la confidentialité des données, consultez la section Confidentialité des données FAQ. Pour plus d'informations sur la protection des données en Europe, consultez le modèle de responsabilitéAWS partagée et le billet de GDPR blog sur le blog sur la AWS sécurité.

À des fins de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer les utilisateurs individuels avec AWS IAM Identity Center ou AWS Identity and Access Management (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multifactorielle (MFA) pour chaque compte.
- UtilisezSSL/TLSpour communiquer avec les AWS ressources. Nous avons besoin de la TLS version 1.2 et recommandons la TLS version 1.3.
- Configuration API et journalisation de l'activité des utilisateurs avec AWS CloudTrail. Pour plus d'informations sur l'utilisation des CloudTrail sentiers pour capturer AWS des activités, consultez la section Utilisation des CloudTrail sentiers dans le guide de AWS CloudTrail l'utilisateur.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut qu'ils contiennent Services AWS.
- Utilisez des services de sécurité gérés avancés tels qu'Amazon Macie, qui contribuent à la découverte et à la sécurisation des données sensibles stockées dans Amazon S3.
- Si vous avez besoin de FIPS 140 à 3 modules cryptographiques validés pour accéder AWS via une interface de ligne de commande ou unAPI, utilisez un point de terminaison. FIPS Pour plus d'informations sur les FIPS points de terminaison disponibles, voir Federal Information Processing Standard (FIPS) 140-3.

Nous vous recommandons fortement de ne jamais placer d'informations confidentielles ou sensibles, telles que les adresses e-mail de vos clients, dans des balises ou des champs de texte libre tels que le champ Name (Nom). Cela inclut lorsque vous travaillez avec ce AWS produit ou service ou autre Services AWS en utilisant la consoleAPI, AWS CLI, ou AWS SDKs. Toutes les données que vous entrez dans des balises ou des champs de texte de forme libre utilisés pour les noms peuvent être utilisées à des fins de facturation ou dans les journaux de diagnostic. Si vous fournissez un URL à un serveur externe, nous vous recommandons vivement de ne pas inclure d'informations d'identification dans le URL afin de valider votre demande auprès de ce serveur.

## Chiffrement des données

Une caractéristique clé de tout service sécurisé est que les informations sont chiffrées lorsqu'elles ne sont pas utilisées activement.

### Chiffrement au repos

Il AWS Tools for PowerShell ne stocke lui-même aucune donnée client autre que les informations d'identification dont il a besoin pour interagir avec les AWS services au nom de l'utilisateur.

Si vous utilisez le AWS Tools for PowerShell pour appeler un AWS service qui transmet les données clients à votre ordinateur local à des fins de stockage, reportez-vous au chapitre Sécurité et conformité du guide de l'utilisateur de ce service pour obtenir des informations sur la manière dont ces données sont stockées, protégées et cryptées.

### Chiffrement en transit

Par défaut, toutes les données transmises depuis l'ordinateur client exécutant les points de terminaison AWS Tools for PowerShell et de AWS service sont cryptées en envoyant le tout via une TLS connexionHTTPS/.

Vous n'avez rien à faire pour activer l'utilisation deHTTPS/TLS. Le protocole est toujours activé.

# Gestion de l’identité et des accès

AWS Identity and Access Management (IAM) est un outil Service AWS qui permet à un administrateur de contrôler en toute sécurité l'accès aux AWS ressources. IAMles administrateurs contrôlent qui peut être authentifié (connecté) et autorisé (autorisé) à utiliser les AWS ressources. IAMest un Service AWS outil que vous pouvez utiliser sans frais supplémentaires.

Rubriques

- Public ciblé
- Authentification par des identités
- Gestion des accès à l’aide de politiques
- Comment Services AWS travailler avec IAM
- Résolution des problèmes AWS d'identité et d'accès

## Public ciblé

La façon dont vous utilisez AWS Identity and Access Management (IAM) varie en fonction du travail que vous effectuez AWS.

Utilisateur du service : si vous avez l' Services AWS habitude de faire votre travail, votre administrateur vous fournit les informations d'identification et les autorisations dont vous avez besoin. Au fur et à mesure que vous utilisez de nouvelles AWS fonctionnalités pour effectuer votre travail, vous aurez peut-être besoin d'autorisations supplémentaires. En comprenant bien la gestion des accès, vous saurez demander les autorisations appropriées à votre administrateur. Si vous ne pouvez pas accéder à une fonctionnalité dans AWS, consultez Résolution des problèmes AWS d'identité et d'accès le guide de l'utilisateur du Service AWS que vous utilisez.

Administrateur du service — Si vous êtes responsable des AWS ressources de votre entreprise, vous avez probablement un accès complet à AWS. C'est à vous de déterminer les AWS fonctionnalités et les ressources auxquelles les utilisateurs de votre service doivent accéder. Vous devez ensuite envoyer des demandes à votre IAM administrateur pour modifier les autorisations des utilisateurs de votre service. Consultez les informations de cette page pour comprendre les concepts de base deIAM. Pour en savoir plus sur la façon dont votre entreprise peut IAM l'utiliser AWS, consultez le guide de l'utilisateur Service AWS que vous utilisez.

IAMadministrateur — Si vous êtes IAM administrateur, vous souhaiterez peut-être en savoir plus sur la manière dont vous pouvez rédiger des politiques pour gérer l'accès à AWS. Pour consulter des exemples de politiques AWS basées sur l'identité que vous pouvez utiliserIAM, consultez le guide de l'utilisateur du logiciel Service AWS que vous utilisez.

## Authentification par des identités

L'authentification est la façon dont vous vous connectez à AWS l'aide de vos informations d'identification. Vous devez être authentifié (connecté à AWS) en tant que Utilisateur racine d'un compte AWS, en tant qu'IAMutilisateur ou en assumant un IAM rôle.

Vous pouvez vous connecter en AWS tant qu'identité fédérée en utilisant les informations d'identification fournies par le biais d'une source d'identité. AWS IAM Identity Center Les utilisateurs (IAMIdentity Center), l'authentification unique de votre entreprise et vos informations d'identification Google ou Facebook sont des exemples d'identités fédérées. Lorsque vous vous connectez en tant qu'identité fédérée, votre administrateur a préalablement configuré la fédération d'identité à l'aide de IAM rôles. Lorsque vous accédez à AWS l'aide de la fédération, vous assumez indirectement un rôle.

Selon le type d'utilisateur que vous êtes, vous pouvez vous connecter au portail AWS Management Console ou au portail AWS d'accès. Pour plus d'informations sur la connexion à AWS, consultez la section Comment vous connecter à votre compte Compte AWS dans le guide de Connexion à AWS l'utilisateur.

Si vous y accédez AWS par programmation, AWS fournit un kit de développement logiciel (SDK) et une interface de ligne de commande (CLI) pour signer cryptographiquement vos demandes à l'aide de vos informations d'identification. Si vous n'utilisez pas d' AWS outils, vous devez signer vous-même les demandes. Pour plus d'informations sur l'utilisation de la méthode recommandée pour signer vous-même les demandes, consultez la section Signature des AWS API demandes dans le guide de IAM l'utilisateur.

Quelle que soit la méthode d'authentification que vous utilisez, vous devrez peut-être fournir des informations de sécurité supplémentaires. Par exemple, il vous AWS recommande d'utiliser l'authentification multifactorielle (MFA) pour renforcer la sécurité de votre compte. Pour en savoir plus, consultez Authentification multifactorielle dans le guide de AWS IAM Identity Center l'utilisateur et Utilisation de l'authentification multifactorielle (MFA) AWS dans le guide de l'IAMutilisateur.

## Compte AWS utilisateur root

Lorsque vous créez un Compte AWS, vous commencez par une identité de connexion unique qui donne un accès complet à toutes Services AWS les ressources du compte. Cette identité est appelée utilisateur Compte AWS root et est accessible en vous connectant avec l'adresse e-mail et le mot de passe que vous avez utilisés pour créer le compte. Il est vivement recommandé de ne pas utiliser l'utilisateur racine pour vos tâches quotidiennes. Protégez vos informations d'identification d'utilisateur racine et utilisez-les pour effectuer les tâches que seul l'utilisateur racine peut effectuer. Pour obtenir la liste complète des tâches qui nécessitent que vous vous connectiez en tant qu'utilisateur root, consultez la section Tâches nécessitant des informations d'identification utilisateur root dans le guide de IAM l'utilisateur.

## Identité fédérée

La meilleure pratique consiste à obliger les utilisateurs humains, y compris ceux qui ont besoin d'un accès administrateur, à utiliser la fédération avec un fournisseur d'identité pour accéder à l'aide Services AWS d'informations d'identification temporaires.

Une identité fédérée est un utilisateur de l'annuaire des utilisateurs de votre entreprise, d'un fournisseur d'identité Web AWS Directory Service, du répertoire Identity Center ou de tout utilisateur qui y accède en utilisant les informations d'identification fournies Services AWS par le biais d'une

source d'identité. Lorsque des identités fédérées y accèdent Comptes AWS, elles assument des rôles, qui fournissent des informations d'identification temporaires.

Pour une gestion des accès centralisée, nous vous recommandons d'utiliser AWS IAM Identity Center. Vous pouvez créer des utilisateurs et des groupes dans IAM Identity Center, ou vous pouvez vous connecter et synchroniser avec un ensemble d'utilisateurs et de groupes dans votre propre source d'identité afin de les utiliser dans toutes vos applications Comptes AWS et applications. Pour plus d'informations sur IAM Identity Center, consultez Qu'est-ce qu'IAMIdentity Center ? dans le guide de AWS IAM Identity Center l'utilisateur.

## Utilisateurs et groupes IAM

Un IAMutilisateur est une identité au sein de vous Compte AWS qui possède des autorisations spécifiques pour une seule personne ou une seule application. Dans la mesure du possible, nous vous recommandons de vous appuyer sur des informations d'identification temporaires plutôt que de créer des IAM utilisateurs dotés d'informations d'identification à long terme, telles que des mots de passe et des clés d'accès. Toutefois, si vous avez des cas d'utilisation spécifiques qui nécessitent des informations d'identification à long terme auprès des IAM utilisateurs, nous vous recommandons de faire pivoter les clés d'accès. Pour plus d'informations, voir Rotation régulière des clés d'accès pour les cas d'utilisation nécessitant des informations d'identification à long terme dans le Guide de IAM l'utilisateur.

Un IAMgroupe est une identité qui définit un ensemble d'IAMutilisateurs. Vous ne pouvez pas vous connecter en tant que groupe. Vous pouvez utiliser les groupes pour spécifier des autorisations pour plusieurs utilisateurs à la fois. Les groupes permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Par exemple, vous pouvez nommer un groupe IAMAdminset lui donner les autorisations nécessaires pour administrer IAM des ressources.

Les utilisateurs sont différents des rôles. Un utilisateur est associé de manière unique à une personne ou une application, alors qu'un rôle est conçu pour être endossé par tout utilisateur qui en a besoin. Les utilisateurs disposent d'informations d'identification permanentes, mais les rôles fournissent des informations d'identification temporaires. Pour en savoir plus, voir Quand créer un IAM utilisateur (au lieu d'un rôle) dans le Guide de IAM l'utilisateur.

## IAMrôles

Un IAMrôle est une identité au sein de Compte AWS vous dotée d'autorisations spécifiques. Il est similaire à un IAM utilisateur, mais n'est pas associé à une personne en particulier. Vous pouvez assumer temporairement un IAM rôle dans le en AWS Management Console changeant de rôle.

Vous pouvez assumer un rôle en appelant une AWS API opération AWS CLI or ou en utilisant une option personnaliséeURL. Pour plus d'informations sur les méthodes d'utilisation des rôles, consultez la section Utilisation IAM des rôles dans le Guide de IAM l'utilisateur.

IAMles rôles dotés d'informations d'identification temporaires sont utiles dans les situations suivantes :

- Accès utilisateur fédéré : pour attribuer des autorisations à une identité fédérée, vous créez un rôle et définissez des autorisations pour le rôle. Quand une identité externe s'authentifie, l'identité est associée au rôle et reçoit les autorisations qui sont définies par celui-ci. Pour plus d'informations sur les rôles pour la fédération, voir Création d'un rôle pour un fournisseur d'identité tiers dans le guide de IAM l'utilisateur. Si vous utilisez IAM Identity Center, vous configurez un ensemble d'autorisations. Pour contrôler les accès auxquels vos identités peuvent accéder après leur authentification, IAM Identity Center met en corrélation l'ensemble d'autorisations avec un rôle dans. IAM Pour plus d'informations sur les jeux d'autorisations, consultez Jeux d'autorisations dans le Guide de l'utilisateur AWS IAM Identity Center .
- Autorisations IAM utilisateur temporaires : un IAM utilisateur ou un rôle peut assumer un IAM rôle afin d'obtenir temporairement différentes autorisations pour une tâche spécifique.
- Accès entre comptes : vous pouvez utiliser un IAM rôle pour autoriser une personne (un mandant fiable) d'un autre compte à accéder aux ressources de votre compte. Les rôles constituent le principal moyen d'accorder l'accès intercompte. Toutefois, dans certains Services AWS cas, vous pouvez associer une politique directement à une ressource (au lieu d'utiliser un rôle comme proxy). Pour connaître la différence entre les rôles et les politiques basées sur les ressources pour l'accès entre comptes, voir Accès aux ressources entre comptes IAM dans le guide de l'IAMutilisateur.
- Accès multiservices — Certains Services AWS utilisent des fonctionnalités dans d'autres Services AWS. Par exemple, lorsque vous effectuez un appel dans un service, il est courant que ce service exécute des applications dans Amazon EC2 ou stocke des objets dans Amazon S3. Un service peut le faire en utilisant les autorisations d'appel du principal, un rôle de service ou un rôle lié au service.
  - Sessions d'accès transmises (FAS) — Lorsque vous utilisez un IAM utilisateur ou un rôle pour effectuer des actions AWS, vous êtes considéré comme un mandant. Lorsque vous utilisez certains services, vous pouvez effectuer une action qui initie une autre action dans un autre service. FASutilise les autorisations du principal appelant an Service AWS, combinées à la demande Service AWS pour adresser des demandes aux services en aval. FASles demandes ne sont effectuées que lorsqu'un service reçoit une demande qui nécessite des interactions avec d'autres personnes Services AWS ou des ressources pour être traitée. Dans ce cas, vous devez

disposer d’autorisations nécessaires pour effectuer les deux actions. Pour plus de détails sur les politiques relatives à l'envoi de FAS demandes, consultez la section Transférer les sessions d'accès.

- Rôle de service — Un rôle de service est un IAMrôle qu'un service assume pour effectuer des actions en votre nom. Un IAM administrateur peut créer, modifier et supprimer un rôle de service de l'intérieurIAM. Pour plus d'informations, consultez la section Création d'un rôle auquel déléguer des autorisations Service AWS dans le Guide de IAM l'utilisateur.
- Rôle lié à un service — Un rôle lié à un service est un type de rôle de service lié à un. Service AWS Le service peut endosser le rôle afin d’effectuer une action en votre nom. Les rôles liés à un service apparaissent dans votre Compte AWS répertoire et appartiennent au service. Un IAM administrateur peut consulter, mais pas modifier les autorisations pour les rôles liés à un service.

- Applications exécutées sur Amazon EC2 : vous pouvez utiliser un IAM rôle pour gérer les informations d'identification temporaires pour les applications qui s'exécutent sur une EC2 instance et qui font AWS CLI des AWS API demandes. Cela est préférable au stockage des clés d'accès dans l'EC2instance. Pour attribuer un AWS rôle à une EC2 instance et le rendre disponible pour toutes ses applications, vous devez créer un profil d'instance attaché à l'instance. Un profil d'instance contient le rôle et permet aux programmes exécutés sur l'EC2instance d'obtenir des informations d'identification temporaires. Pour plus d'informations, consultez la section Utilisation d'un IAM rôle pour accorder des autorisations aux applications exécutées sur des EC2 instances Amazon dans le Guide de IAM l'utilisateur.

Pour savoir s'il faut utiliser IAM des rôles ou des IAM utilisateurs, voir Quand créer un IAM rôle (au lieu d'un utilisateur) dans le guide de IAM l'utilisateur.

## Gestion des accès à l’aide de politiques

Vous contrôlez l'accès en AWS créant des politiques et en les associant à AWS des identités ou à des ressources. Une politique est un objet AWS qui, lorsqu'il est associé à une identité ou à une ressource, définit leurs autorisations. AWS évalue ces politiques lorsqu'un principal (utilisateur, utilisateur root ou session de rôle) fait une demande. Les autorisations dans les politiques déterminent si la demande est autorisée ou refusée. La plupart des politiques sont stockées AWS sous forme de JSON documents. Pour plus d'informations sur la structure et le contenu des documents de JSON politique, voir Présentation des JSON politiques dans le guide de IAM l'utilisateur.

Les administrateurs peuvent utiliser AWS JSON des politiques pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

Par défaut, les utilisateurs et les rôles ne disposent d'aucune autorisation. Pour autoriser les utilisateurs à effectuer des actions sur les ressources dont ils ont besoin, un IAM administrateur peut créer des IAM politiques. L'administrateur peut ensuite ajouter les IAM politiques aux rôles, et les utilisateurs peuvent assumer les rôles.

IAMles politiques définissent les autorisations pour une action, quelle que soit la méthode que vous utilisez pour effectuer l'opération. Par exemple, supposons que vous disposiez d'une politique qui autorise l'action `iam:GetRole`. Un utilisateur doté de cette politique peut obtenir des informations sur le rôle auprès du AWS Management Console AWS CLI, ou du AWS API.

## Politiques basées sur l'identité

Les politiques basées sur l'identité sont JSON des documents de politique d'autorisation que vous pouvez joindre à une identité, telle qu'un IAM utilisateur, un groupe d'utilisateurs ou un rôle. Ces politiques contrôlent quel type d'actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour savoir comment créer une politique basée sur l'identité, consultez la section Création de IAM politiques dans le Guide de l'IAMutilisateur.

Les politiques basées sur l'identité peuvent être classées comme des politiques en ligne ou des politiques gérées. Les politiques en ligne sont intégrées directement à un utilisateur, groupe ou rôle. Les politiques gérées sont des politiques autonomes que vous pouvez associer à plusieurs utilisateurs, groupes et rôles au sein de votre Compte AWS. Les politiques gérées incluent les politiques AWS gérées et les politiques gérées par le client. Pour savoir comment choisir entre une politique gérée ou une politique intégrée, voir Choisir entre des politiques gérées et des politiques intégrées dans le Guide de l'IAMutilisateur.

## Politiques basées sur les ressources

Les politiques basées sur les ressources sont des documents JSON de stratégie que vous attachez à une ressource. Les politiques de confiance dans les IAM rôles et les politiques relatives aux compartiments Amazon S3 sont des exemples de politiques basées sur les ressources. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez spécifier un principal dans une

politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou. Services AWS

Les politiques basées sur les ressources sont des politiques en ligne situées dans ce service. Vous ne pouvez pas utiliser de politiques AWS gérées depuis une IAM stratégie basée sur les ressources.

### Listes de contrôle d'accès (ACLs)

Les listes de contrôle d'accès (ACLs) contrôlent les principaux (membres du compte, utilisateurs ou rôles) autorisés à accéder à une ressource. ACLssont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format du document JSON de stratégie.

Amazon S3 et Amazon VPC sont des exemples de services qui prennent en chargeACLs. AWS WAF Pour en savoir plusACLs, consultez la présentation de la liste de contrôle d'accès (ACL) dans le guide du développeur Amazon Simple Storage Service.

### Autres types de politique

AWS prend en charge d'autres types de politiques moins courants. Ces types de politiques peuvent définir le nombre maximum d’autorisations qui vous sont accordées par des types de politiques plus courants.

- Limites d'autorisations — Une limite d'autorisations est une fonctionnalité avancée dans laquelle vous définissez le maximum d'autorisations qu'une politique basée sur l'identité peut accorder à une IAM entité (IAMutilisateur ou rôle). Vous pouvez définir une limite d’autorisations pour une entité. Les autorisations en résultant représentent la combinaison des politiques basées sur l’identité d’une entité et de ses limites d’autorisation. Les politiques basées sur les ressources qui spécifient l’utilisateur ou le rôle dans le champ `Principal` ne sont pas limitées par les limites d’autorisations. Un refus explicite dans l’une de ces politiques annule l’autorisation. Pour plus d'informations sur les limites d'autorisations, consultez la section Limites d'autorisations pour les IAM entités dans le Guide de IAM l'utilisateur.
- Politiques de contrôle des services (SCPs) : SCPs JSON politiques qui spécifient les autorisations maximales pour une organisation ou une unité organisationnelle (UO) dans AWS Organizations. AWS Organizations est un service permettant de regrouper et de gérer de manière centralisée Comptes AWS les multiples propriétés de votre entreprise. Si vous activez toutes les fonctionnalités d'une organisation, vous pouvez appliquer des politiques de contrôle des services (SCPs) à l'un ou à l'ensemble de vos comptes. Les SCP limites d'autorisations pour les entités présentes dans les comptes des membres, y compris chacune d'entre elles Utilisateur racine d'un

compte AWS. Pour plus d'informations sur les Organizations et consultez SCPs les politiques de contrôle des services dans le Guide de AWS Organizations l'utilisateur.

- Politiques de séance : les politiques de séance sont des politiques avancées que vous utilisez en tant que paramètre lorsque vous créez par programmation une séance temporaire pour un rôle ou un utilisateur fédéré. Les autorisations de séance en résultant sont une combinaison des politiques basées sur l'identité de l'utilisateur ou du rôle et des politiques de séance. Les autorisations peuvent également provenir d'une politique basée sur les ressources. Un refus explicite dans l'une de ces politiques annule l'autorisation. Pour plus d'informations, consultez la section Politiques de session dans le guide de IAM l'utilisateur.

### Plusieurs types de politique

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations en résultant sont plus compliquées à comprendre. Pour savoir comment AWS déterminer s'il faut autoriser une demande lorsque plusieurs types de politiques sont impliqués, consultez la section Logique d'évaluation des politiques dans le guide de IAM l'utilisateur.

## Comment Services AWS travailler avec IAM

Pour obtenir une vue d'ensemble du Services AWS fonctionnement de la plupart des IAM fonctionnalités, consultez AWS les services compatibles IAM dans le Guide de IAM l'utilisateur.

Pour savoir comment utiliser un outil spécifique Service AWS IAM, consultez la section relative à la sécurité du guide de l'utilisateur du service concerné.

## Résolution des problèmes AWS d'identité et d'accès

Utilisez les informations suivantes pour vous aider à diagnostiquer et à résoudre les problèmes courants que vous pouvez rencontrer lorsque vous travaillez avec AWS etIAM.

Rubriques

- Je ne suis pas autorisé à effectuer une action dans AWS
- Je ne suis pas autorisé à effectuer iam : PassRole
- Je souhaite permettre à des personnes extérieures Compte AWS à moi d'accéder à mes AWS ressources

## Je ne suis pas autorisé à effectuer une action dans AWS

Si vous recevez une erreur qui indique que vous n'êtes pas autorisé à effectuer une action, vos politiques doivent être mises à jour afin de vous permettre d'effectuer l'action.

L'exemple d'erreur suivant se produit lorsque l'`mateojackson`IAMutilisateur essaie d'utiliser la console pour afficher les détails d'une *`my-example-widget`* ressource fictive mais ne dispose pas des `awes:`*`GetWidget`* autorisations fictives.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes:GetWidget on resource: my-example-widget
```

Dans ce cas, la politique qui s'applique à l'utilisateur `mateojackson` doit être mise à jour pour autoriser l'accès à la ressource *`my-example-widget`* à l'aide de l'action `awes:`*`GetWidget`*.

Si vous avez besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d'identification de connexion.

## Je ne suis pas autorisé à effectuer iam : PassRole

Si vous recevez une erreur selon laquelle vous n'êtes pas autorisé à exécuter `iam:PassRole` l'action, vos stratégies doivent être mises à jour afin de vous permettre de transmettre un rôle à AWS.

Certains vous Services AWS permettent de transmettre un rôle existant à ce service au lieu de créer un nouveau rôle de service ou un rôle lié à un service. Pour ce faire, un utilisateur doit disposer des autorisations nécessaires pour transmettre le rôle au service.

L'exemple d'erreur suivant se produit lorsqu'un IAM utilisateur nommé `marymajor` essaie d'utiliser la console pour effectuer une action dans AWS. Toutefois, l'action nécessite que le service ait des autorisations accordées par un rôle de service. Mary ne dispose pas des autorisations nécessaires pour transférer le rôle au service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dans ce cas, les politiques de Mary doivent être mises à jour pour lui permettre d'exécuter l'action `iam:PassRole`.

Si vous avez besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d'identification de connexion.

### Je souhaite permettre à des personnes extérieures Compte AWS à moi d'accéder à mes AWS ressources

Vous pouvez créer un rôle que les utilisateurs provenant d'autres comptes ou les personnes extérieures à votre organisation pourront utiliser pour accéder à vos ressources. Vous pouvez spécifier qui est autorisé à assumer le rôle. Pour les services qui prennent en charge les politiques basées sur les ressources ou les listes de contrôle d'accès (ACLs), vous pouvez utiliser ces politiques pour autoriser les utilisateurs à accéder à vos ressources.

Pour en savoir plus, consultez les éléments suivants :

- Pour savoir si ces fonctionnalités sont prises AWS en charge, consultezComment Services AWS travailler avec IAM.
- Pour savoir comment donner accès à vos ressources sur un site Comptes AWS qui vous appartient, consultez la section Fournir l'accès à un IAM utilisateur dans un autre site Compte AWS que vous possédez dans le Guide de IAM l'utilisateur.
- Pour savoir comment fournir l'accès à vos ressources à des tiers Comptes AWS, consultez la section Fournir un accès à des ressources Comptes AWS détenues par des tiers dans le Guide de IAM l'utilisateur.
- Pour savoir comment fournir un accès via la fédération d'identité, consultez la section Fournir un accès aux utilisateurs authentifiés de manière externe (fédération d'identité) dans le guide de l'IAMutilisateur.
- Pour connaître la différence entre l'utilisation de rôles et l'utilisation de politiques basées sur les ressources pour l'accès entre comptes, voir Accès aux ressources entre comptes IAM dans le guide de l'IAMutilisateur.

## Validation de conformité pour ce AWS produit ou service

Pour savoir si un programme Services AWS de conformité Service AWS s'inscrit dans le champ d'application de programmes de conformité spécifiques, consultez Services AWS la section de conformité et sélectionnez le programme de conformité qui vous intéresse. Pour des informations générales, voir Programmes de AWS conformité Programmes AWS de .

Vous pouvez télécharger des rapports d'audit tiers à l'aide de AWS Artifact. Pour plus d'informations, voir Téléchargement de rapports dans AWS Artifact .

Votre responsabilité en matière de conformité lors de l'utilisation Services AWS est déterminée par la sensibilité de vos données, les objectifs de conformité de votre entreprise et les lois et réglementations applicables. AWS fournit les ressources suivantes pour faciliter la mise en conformité :

- Guides de démarrage rapide sur la sécurité et la conformité : ces guides de déploiement abordent les considérations architecturales et indiquent les étapes à suivre pour déployer des environnements de base axés sur AWS la sécurité et la conformité.
- Architecture axée sur la HIPAA sécurité et la conformité sur Amazon Web Services : ce livre blanc décrit comment les entreprises peuvent AWS créer HIPAA des applications éligibles.

> **Note**
>
> Tous ne Services AWS sont pas HIPAA éligibles. Pour plus d'informations, consultez la référence des services HIPAA éligibles.

- AWS Ressources de https://aws.amazon.com/compliance/resources/ de conformité — Cette collection de classeurs et de guides peut s'appliquer à votre secteur d'activité et à votre région.
- AWS Guides de conformité destinés aux clients — Comprenez le modèle de responsabilité partagée sous l'angle de la conformité. Les guides résument les meilleures pratiques en matière de sécurisation Services AWS et reprennent les directives relatives aux contrôles de sécurité dans de nombreux cadres (notamment le National Institute of Standards and Technology (NIST), le Payment Card Industry Security Standards Council (PCI) et l'Organisation internationale de normalisation (ISO)).
- Évaluation des ressources à l'aide des règles du guide du AWS Config développeur : le AWS Config service évalue dans quelle mesure les configurations de vos ressources sont conformes aux pratiques internes, aux directives du secteur et aux réglementations.
- AWS Security Hub— Cela Service AWS fournit une vue complète de votre état de sécurité interne AWS. Security Hub utilise des contrôles de sécurité pour évaluer vos ressources AWS et vérifier votre conformité par rapport aux normes et aux bonnes pratiques du secteur de la sécurité. Pour obtenir la liste des services et des contrôles pris en charge, consultez Référence des contrôles Security Hub.
- Amazon GuardDuty — Cela Service AWS détecte les menaces potentielles qui pèsent sur vos charges de travail Comptes AWS, vos conteneurs et vos données en surveillant votre environnement pour détecter toute activité suspecte et malveillante. GuardDuty peut vous aider à

répondre à diverses exigences de conformité PCIDSS, par exemple en répondant aux exigences de détection des intrusions imposées par certains cadres de conformité.

- AWS Audit Manager— Cela vous Service AWS permet d'auditer en permanence votre AWS utilisation afin de simplifier la gestion des risques et la conformité aux réglementations et aux normes du secteur.

Ce AWS produit ou service suit le modèle de responsabilité partagée par le biais des services Amazon Web Services (AWS) spécifiques qu'il prend en charge. Pour obtenir des informations sur la sécurité des AWS services, consultez la AWS page de documentation sur la sécuritéAWS des services et les services concernés par les efforts de AWS conformité par programme de conformité.

# Application d'une version minimale de TLS dans Tools for PowerShell

Pour renforcer la sécurité lors de la communication avec les services AWS, il est recommandé de configurer Tools for PowerShell de sorte à utiliser la version appropriée de TLS. Pour plus d'informations sur la procédure à suivre, consultez Enforcing a minimum TLS version dans le manuel AWS SDK for .NET Developer Guide.

# Considérations de sécurité supplémentaires pour les outils pour PowerShell

Cette rubrique contient des considérations relatives à la sécurité en plus des sujets de sécurité abordés dans les sections précédentes.

## Enregistrement d'informations sensibles

Certaines opérations de cet outil peuvent renvoyer des informations susceptibles d'être considérées comme sensibles, notamment des informations provenant de variables d'environnement. L'exposition de ces informations peut représenter un risque de sécurité dans certains scénarios ; par exemple, les informations peuvent être incluses dans les journaux d'intégration continue et de déploiement continu (CI/CD). Il est donc important de vérifier à quel moment vous incluez une telle sortie dans vos journaux et de supprimer la sortie lorsqu'elle n'est pas nécessaire. Pour plus d'informations sur la protection des données sensibles, consultezProtection des données dans ce AWS produit ou service.

Tenez compte des meilleures pratiques suivantes :

- N'utilisez pas de variables d'environnement pour stocker des valeurs sensibles pour vos ressources sans serveur. Demandez plutôt à votre code sans serveur de récupérer le secret par programmation dans un magasin de secrets (par exemple,). AWS Secrets Manager
- Vérifiez le contenu de vos journaux de compilation pour vous assurer qu'ils ne contiennent pas d'informations sensibles. Envisagez des approches telles que le transfert vers /dev/null ou la capture de la sortie sous forme de bash ou de PowerShell variable pour supprimer les sorties de commande.
- Tenez compte de l'accès à vos journaux et délimitez cet accès en fonction de votre cas d'utilisation.

# Références pour les applets de commande Tools for PowerShell

Tools for PowerShell fournit des applets de commande que vous pouvez utiliser pour accéder aux services AWS. Pour déterminer les applets de commande disponibles, consultez la référence des applets de commande AWS Tools for PowerShell.

# Historique du document

Cette rubrique décrit les modifications importantes apportées à la documentation relative aux AWS Tools for PowerShell.

Nous mettons aussi régulièrement à jour la documentation en fonction des commentaires des clients. Pour envoyer des commentaires sur une rubrique, utilisez les boutons de commentaire situés en regard de « Cette page vous a-t-elle été utile ? » en bas de chaque page.

Pour plus d'informations sur les modifications et les mises à jour apportées au AWS Tools for PowerShell, consultez les notes de publication.

| Modification | Description | Date |
| --- | --- | --- |
| Observabilité | Ajout d'informations d'aperçu sur l'observabilité dans le AWS Tools for PowerShell, qui permettent de collecter des données de télémétrie. | 13 septembre 2024 |
| Installation du AWS Tools for PowerShell sous Windows | Ajout d'informations sur le déblocage ZIP des fichiers avant d'en extraire le contenu. | 5 août 2024 |
| Informations sur EC2 -Classic | Suppression des informations concernant EC2 -Classic, qui ont été supprimées. | 1er août 2024 |
| Exemples de code | Un chapitre contenant des exemples d'applets de commande a été inclus. | 17 avril 2024 |
| Considérations de sécurité supplémentaires | Informations incluses sur l'enregistrement potentiel de données sensibles. | 16 avril 2024 |

| | | |
|---|---|---|
| Configurez l'authentification des outils avec AWS | Ajout d'informations sur la prise SSO en charge de AWS Tools for PowerShell. | 15 mars 2024 |
| Référence des applets de commande pour les outils pour PowerShell | Ajout d'une section avec un lien vers les outils de référence pour les PowerShell applets de commande. | 17 novembre 2023 |
| Inclut d'autres mises à jour des IAM meilleures pratiques | Guide mis à jour pour s'aligner sur les IAM meilleures pratiques. Pour plus d'informations, consultez la section Bonnes pratiques en matière de sécurité dans IAM. | 12 octobre 2023 |
| Installation sur Windows | Suppression des informations relatives à l'installation des outils pour Windows PowerShell à l'aide duMSI, qui est devenue obsolète. | 25 septembre 2023 |
| IAMmises à jour des meilleures pratiques | Guide mis à jour pour s'aligner sur les IAM meilleures pratiques. Pour plus d'informations, consultez la section Bonnes pratiques en matière de sécurité dans IAM. | 8 septembre 2023 |
| Pipelining et $ AWSHistory | Le paramètre `IncludeSensitiveData` a été ajouté à l'applet de commande `Set-AWSHistoryConfiguration` . | 9 mars 2023 |
| Utilisation du ClientConfig paramètre dans les applets de commande | Ajout d'informations sur la prise en charge du ClientConfig paramètre. | 28 octobre 2022 |

| | | |
|---|---|---|
| Lancer une EC2 instance Amazon à l'aide de Windows PowerShell | Ajout de notes concernant le retrait de EC2 -Classic. | 26 juillet 2022 |
| AWS Tools for PowerShell Version 4 | Ajout d'informations sur la version 4, y compris des instructions d'installation pour Windows et Linux/mac OS, ainsi qu'une rubrique sur la migration qui décrit les différences par rapport à la version 3 et présente les nouvelles fonctionnalités. | 21 novembre 2019 |
| AWS Tools for PowerShell 3,3,563 | Ajout d'informations sur l'installation et l'utilisation de la version préliminaire du module `AWS.Tools.Common` . Ce nouveau module divise l'ancien package monolithique en un module partagé et un module par AWS service. | 18 octobre 2019 |
| AWS Tools for PowerShell 3.3.343.0 | Ajout d'informations à la section Utilisation de la AWS Tools for PowerShell section présentant les AWS Lambda outils PowerShell permettant aux développeurs PowerShell principaux de créer des AWS Lambda fonctions. | 11 septembre 2018 |

| | | |
|---|---|---|
| AWS Tools for Windows PowerShell 3.1.31.0 | Ajout d'informations à la section Getting Started concernant les nouvelles applets de commande qui utilisent le langage de balisage d'assertions de sécurité (SAML) pour prendre en charge la configuration de l'identité fédérée pour les utilisateurs. | 1 décembre 2015 |
| AWS Tools for Windows PowerShell 2.3,19 | Ajout d'informations à la section Découverte des applets de commande et alias concernant la nouvelle `Get-AWSCmdletName` applet de commande qui peuvent aider les utilisateurs à trouver plus facilement les applets de commande souhaités. AWS | 5 février 2015 |

| | | |
|---|---|---|
| AWS Tools for Windows PowerShell 1.1.1.0 | La sortie de collection des applets de commande est toujours énumérée dans le pipeline. PowerShell Prise en charge automatique des appels de service paginable s. La nouvelle variable AWSHistory shell $ collecte les réponses de service et éventuellement les demandes de service. AWSRegionles instances utilisent le champ Region plutôt SystemName que pour faciliter le pipeline. Remove-S3Bucketprend en charge une option - DeleteObj ects switch. Correction d'un problème d'utilisabilité avec Set-AWSCredentials. Initialis er : AWSDefaults indique d'où il a obtenu les informati ons d'identification et les données régionales. Stop-EC2Instanceaccepte Amazon. EC2Instances .Model.Re servation en entrée. Les types de paramètre de liste générique <T> ont été remplacés par les types de tableau (T[]). Applets de commande qui supprimen t ou résilient l'invite des ressources pour confirmation avant la suppression. Write-S3Objectprend en charge | 15 mai 2013 |

le contenu texte en ligne à
télécharger sur Amazon S3.

| | | |
|---|---|---|
| AWS Tools for Windows PowerShell 1.0.1.0 | L'emplacement d'installation du PowerShell module Outils pour Windows a été modifié afin que les environnements utilisant Windows PowerShell version 3 puissent tirer parti du chargement automatique. Le module et les fichiers de support sont désormais installés dans un sous-dossier `AWSPowerShell` sous `AWS ToolsPowerShell`. Les fichiers des versions précédentes qui existent dans le dossier `AWS ToolsPowerShell` sont automatiquement supprimés par le programme d'installation. `PSModulePath` Pour Windows PowerShell (toutes les versions) est mis à jour dans cette version pour contenir le dossier parent du module (`AWS ToolsPowerShell`). Pour les systèmes dotés de Windows PowerShell version 2, le raccourci du menu Démarrer est mis à jour pour importer le module depuis le nouvel emplacement, puis l'exécute `rInitialize-AWSDefaults`. Pour les systèmes dotés de Windows PowerShell version 3, le raccourci du menu Démarrer est mis à jour | 21 décembre 2012 |

pour supprimer la `Import-Module` commande, en laissant juste`Initialize-AWSDefaults` . Si vous avez modifié votre PowerShell profil pour exécuter l'un `Import-Module` des `AWSPowerShell.psd1` fichiers, vous devez le mettre à jour pour qu'il pointe vers le nouvel emplacement du fichier (ou, si vous utilisez la PowerShell version 3, supprimer l'`Import-Module` instruction car elle n'est plus nécessaire). À la suite de ces modifications, le PowerShell module Outils pour Windows est désormais répertorié comme module disponible lors de l'exécution`Get-Module -ListAvailable` . En outre, pour les utilisateurs de Windows PowerShell version 3, l'exécution de toute applet de commande exportée par le module chargera automatiquement le module dans le PowerShell shell actuel sans qu'il soit nécessaire de l'utiliser `Import-Module` au préalable. Cela permet l'utilisation interactive des applets de commande sur un système avec une politique d'exécuti

| | | |
|---|---|---|
| | on qui empêche l'exécution de scripts. | |
| AWS Tools for Windows PowerShell 1.0.0.0 | Première version | 6 décembre 2012 |

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.