Leitfaden

AWS Tools for PowerShell

AWS Tools for PowerShell: Leitfaden

Table of Contents

Was sind die AWS Tools for PowerShell?

Dabei AWS Tools for PowerShell handelt es sich um eine Reihe von PowerShell Modulen, die auf der Funktionalität von aufbauen AWS SDK for .NET. Sie AWS Tools for PowerShell ermöglichen es Ihnen, Operationen auf Ihren AWS Ressourcen über die PowerShell Befehlszeile per Skript auszuführen.

Die Cmdlets bieten eine idiomatische Oberfläche PowerShell für die Angabe von Parametern und die Verarbeitung von Ergebnissen, obwohl sie mithilfe der verschiedenen AWS Dienst-HTTP-Abfrage-APIs implementiert werden. Die Cmdlets für die Cmdlets AWS Tools for PowerShell unterstützen beispielsweise Pipelining, PowerShell d. h. Sie können Objekte über die Pipeline in die Cmdlets ein- und ausleiten. PowerShell

Sie AWS Tools for PowerShell sind flexibel in der Art und Weise, wie Sie mit Anmeldeinformationen umgehen können, einschließlich der Unterstützung für die (IAM-) Infrastruktur. AWS Identity and Access Management Sie können die Tools mit IAM-Benutzeranmeldeinformationen, temporären Sicherheitstoken und IAM-Rollen verwenden.

Sie AWS Tools for PowerShell unterstützen dieselben Dienste und AWS Regionen, die vom SDK unterstützt werden. Sie können das AWS Tools for PowerShell auf Computern mit Windows-, Linux- oder MacOS-Betriebssystemen installieren.

> **Note**
>
> AWS Tools for PowerShell Version 4 ist die neueste Hauptversion und ein abwärtskompatibles Update auf AWS Tools for PowerShell Version 3.3. Es wurde um erhebliche Verbesserungen ergänzt, gleichzeitig bleibt das vorhandene Cmdlet-Verhalten beibehalten. Ihre vorhandenen Skripts sollten nach dem Upgrade auf die neue Version weiterhin funktionieren. Wir empfehlen jedoch, sie vor dem Aktualisieren gründlich zu testen. Weitere Hinweise zu den Änderungen in Version 4 finden Sie unter Migrieren von der AWS Tools for PowerShell-Version 3.3 zu Version 4.

Sie AWS Tools for PowerShell sind in den folgenden drei unterschiedlichen Paketen erhältlich:

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

Wartung und Support für SDK-Hauptversionen

Informationen zu Wartung und Support für SDK-Hauptversionen und deren zugrunde liegende Abhängigkeiten finden Sie im AWS -Referenzhandbuch zu SDKs und Tools:

- AWS Wartungsrichtlinien für SDKs und Tools
- AWS Matrix zur Unterstützung der Versionen von SDKs und Tools

AWS.Tools- Eine modularisierte Version des AWS Tools for PowerShell

Diese Version von AWS Tools for PowerShell ist die empfohlene Version für jeden Computer, der PowerShell in einer Produktionsumgebung ausgeführt wird. Da es modularisiert ist, müssen Sie nur die Module für die Services herunterladen und laden, die Sie verwenden möchten. Dies reduziert die Download-Zeiten sowie die Speicherbelegung und ermöglicht den automatischen Import von `AWS.Tools`-Cmdlets, wobei `Import-Module` zuerst manuell aufgerufen werden muss.

Dies ist die neueste Version von AWS Tools for PowerShell und läuft auf allen unterstützten Betriebssystemen, einschließlich Windows, Linux und macOS. Dieses Paket enthält ein Installationsmodul`AWS.Tools.Installer`, ein gemeinsames Modul und ein Modul für jeden AWS Dienst `AWS.Tools.EC2``AWS.Tools.IdentityManagement`, z. B.`AWS.Tools.S3`,, usw. `AWS.Tools.Common`

Das `AWS.Tools.Installer` Modul stellt Cmdlets bereit, mit denen Sie die Module für jeden der Dienste installieren, aktualisieren und entfernen können. AWS Die Cmdlets in diesem Modul stellen automatisch sicher, dass Sie über alle abhängigen Module verfügen, die zur Unterstützung der von Ihnen gewünschten Module erforderlich sind.

Das Modul `AWS.Tools.Common` stellt Cmdlets für Konfiguration und Authentifizierung bereit, die nicht servicespezifisch sind. Um die Cmdlets für einen AWS Dienst zu verwenden, führen Sie einfach den Befehl aus. PowerShellimportiert automatisch das `AWS.Tools.Common` Modul und das Modul für den AWS Dienst, dessen Cmdlet Sie ausführen möchten. Dieses Modul wird automatisch

installiert, wenn Sie das `AWS.Tools.Installer`-Modul zur Installation der Servicemodule verwenden.

Sie können diese Version von AWS Tools for PowerShell auf Computern installieren, auf denen Folgendes ausgeführt wird:

- PowerShell Core 6.0 oder höher unter Windows, Linux oder macOS.
- Windows PowerShell 5.1 oder höher unter Windows mit dem.NET Framework 4.7.2 oder höher.

Wenn wir in diesem Handbuch nur diese Version angeben müssen, verweisen wir darauf mit seinem Modulnamen: *`AWS.Tools`*.

AWSPowerShell. NetCore - Eine Einzelmodulversion des AWS Tools for PowerShell

Diese Version besteht aus einem einzigen, großen Modul, das Unterstützung für alle AWS Dienste enthält. Bevor Sie dieses Modul verwenden können, müssen Sie es manuell importieren.

Sie können diese Version von AWS Tools for PowerShell auf Computern installieren, auf denen Folgendes ausgeführt wird:

- PowerShell Core 6.0 oder höher unter Windows, Linux oder macOS.
- Windows PowerShell 3.0 oder höher unter Windows mit dem.NET Framework 4.7.2 oder höher.

Wenn wir in diesem Handbuch nur diese Version angeben müssen, verweisen wir auf sie mit ihrem Modulnamen:AWSPowerShell. NetCore.

AWSPowerShell - Eine Einzelmodulversion für Windows PowerShell

Diese Version von AWS Tools for PowerShell ist nur mit Windows-Computern kompatibel und kann nur auf Windows-Computern installiert werden, auf denen die PowerShell Windows-Versionen 2.0 bis 5.1 ausgeführt werden. Es ist nicht kompatibel mit PowerShell Core 6.0 oder höher oder einem anderen Betriebssystem (Linux oder macOS). Diese Version besteht aus einem einzigen, großen Modul, das Unterstützung für alle AWS Dienste enthält.

Wenn wir in diesem Handbuch nur diese Version angeben müssen, verweisen wir auf sie mit ihrem Modulnamen: AWSPowerShell.

Verwendung dieses Leitfadens

Das Handbuch ist in die folgenden Abschnitte unterteilt:

Installieren des AWS Tools for PowerShell

In diesem Abschnitt wird erklärt, wie Sie das installieren AWS Tools for PowerShell. Darin erfahren Sie, wie Sie sich registrieren, AWS falls Sie noch kein Konto haben, und wie Sie einen IAM-Benutzer erstellen, mit dem Sie die Cmdlets ausführen können.

Erste Schritte mit der AWS Tools for Windows PowerShell

In diesem Abschnitt werden die Grundlagen der Verwendung von beschrieben AWS Tools for PowerShell, z. B. das Angeben von Anmeldeinformationen und AWS Regionen, das Suchen nach Cmdlets für einen bestimmten Dienst und das Verwenden von Aliasen für Cmdlets.

Arbeiten mit AWS-Services in AWS Tools for PowerShell

Dieser Abschnitt enthält Informationen zur Verwendung von, AWS Tools for PowerShell um einige der häufigsten Aufgaben auszuführen. AWS

Installieren des AWS Tools for PowerShell

Informationen zur erfolgreichen Installation und Verwendung der AWS Tools for PowerShell-Cmdlets finden Sie in den folgenden Themen.

Themen

Installieren der AWS Tools for PowerShell unter Windows

Ein Windows-basierter Computer kann eine der folgenden AWS Tools for PowerShell Paketoptionen ausführen:

- **`AWS.Tools`** – Die modularisierte Version von AWS Tools for PowerShell. Jeder AWS Service wird von einem eigenen, kleinen Modul mit gemeinsamen Supportmodulen `AWS.Tools.Common` und unterstütztAWS.Tools.Installer.
- AWSPowerShell.NetCore – Die einzelne, Large-Modul-Version von AWS Tools for PowerShell. Alle - AWS Services werden von diesem einzigen, großen Modul unterstützt.

  > **Note**
  >
  > Beachten Sie, dass das einzelne Modul möglicherweise zu groß ist, um es mit AWS Lambda-Features zu verwenden. Verwenden Sie stattdessen die oben gezeigte modularisierte Version.

- AWSPowerShell – Die ältere Windows-spezifische große Einzelmodulversion von AWS Tools for PowerShell. Alle - AWS Services werden von diesem einzigen, großen Modul unterstützt.

Das ausgewählte Paket hängt von der Version und Edition von Windows ab, die Sie ausführen.

> **Note**
>
> Die Tools for Windows PowerShell (AWSPowerShell Modul) sind standardmäßig auf allen Windows-basierten Amazon Machine Images (AMIs) installiert.

Das Einrichten von AWS Tools for PowerShell umfasst die folgenden allgemeinen Aufgaben, die in diesem Thema ausführlich beschrieben werden.

1. Installieren Sie die AWS Tools for PowerShell Paketoption, die für Ihre Umgebung geeignet ist.
2. Sicherstellen, dass die Skriptausführung aktiviert ist, indem Sie das Cmdlet `Get-ExecutionPolicy` ausführen
3. Importieren Sie das AWS Tools for PowerShell Modul in Ihre PowerShell Sitzung.

Voraussetzungen

Neuere Versionen von PowerShell, einschließlich PowerShell Core, sind als Downloads von Microsoft unter Installieren verschiedener Versionen von PowerShell auf der Microsoft-Website verfügbar.

Installieren von **`AWS.Tools`** unter Windows.

Sie können die modularisierte Version von AWS Tools for PowerShell auf Computern installieren, auf denen Windows mit Windows PowerShell 5.1 oder PowerShell Core 6.0 oder höher ausgeführt wird. Informationen zur Installation PowerShell von Core finden Sie unter Installieren verschiedener Versionen von PowerShell auf der Microsoft-Website.

Sie können `AWS.Tools` auf drei Arten installieren:

- Verwenden der Cmdlets im `AWS.Tools.Installer`-Modul. Dieses Modul vereinfacht die Installation und Aktualisierung anderer `AWS.Tools` Module. `AWS.Tools.Installer` erfordert `PowerShellGet`und lädt automatisch eine aktualisierte Version davon herunter und installiert sie. hält Ihre Modulversionen `AWS.Tools.Installer` automatisch synchron. Wenn Sie eine neuere Version eines Moduls installieren oder aktualisieren, aktualisieren die Cmdlets in `AWS.Tools.Installer` automatisch alle Ihre anderen `AWS.Tools` Module auf dieselbe Version.

  Diese Methode wird im folgenden Verfahren beschrieben.

- Laden Sie die Module von `AWS.Tools.zip` herunter und extrahieren Sie sie in einem der Modulordner. Sie können Ihre Modulordner erkennen, indem Sie den Wert der `PSModulePath`-Umgebungsvariablen anzeigen.
- Installieren jedes Servicemoduls aus der - PowerShell Galerie mithilfe des `Install-Module` -Cmdlets.

So installieren Sie **`AWS.Tools`** unter Windows mit dem -**`AWS.Tools.Installer`**Modul

1. Starten Sie eine PowerShell Sitzung.

   > **Note**
   >
   > Wir empfehlen, nicht PowerShell als Administrator mit erhöhten Berechtigungen auszuführen, es sei denn, dies ist für die jeweilige Aufgabe erforderlich. Grund ist das potenzielle Sicherheitsrisiko und weil dies im Widerspruch zum Prinzip der geringsten Zugriffsrechte stünde.

2. Führen Sie den folgenden Befehl aus, um das modularisierte `AWS.Tools`-Paket zu installieren.

   ```
   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
   ```

   Wenn Sie benachrichtigt werden, dass das Repository "nicht vertrauenswürdig" ist, werden Sie gefragt, ob die Installation trotzdem erfolgen soll. Geben Sie ein**y**, um die Installation des Moduls PowerShell zu erlauben. Um die Eingabeaufforderung zu vermeiden und das Modul zu installieren, ohne dem Repository zu vertrauen, können Sie den Befehl mit dem `-Force`-Parameter ausführen:

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

3. Sie können jetzt das Modul für jeden AWS Service installieren, den Sie verwenden möchten, indem Sie das `Install-AWSToolsModule` -Cmdlet verwenden. Mit dem folgenden Befehl

werden beispielsweise die Module Amazon EC2 und Amazon S3 installiert. Mit diesem Befehl werden auch alle abhängigen Module installiert, die für die Funktionsfähigkeit des angegebenen Moduls erforderlich sind. Wenn Sie beispielsweise Ihr erstes `AWS.Tools`-Servicemodul installieren, wird damit auch `AWS.Tools.Common` installiert. Dies ist ein gemeinsam genutztes Modul, das von allen AWS Servicemodulen benötigt wird. Damit werden auch ältere Versionen der Module entfernt und weitere Module auf dieselbe neuere Version aktualisiert.

```
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**
>
> Über das Cmdlet `Install-AWSToolsModule` werden alle angeforderten Module aus dem `PSRepository` namens `PSGallery` heruntergeladen (https://www.powershellgallery.com/) und als vertrauenswürdige Quelle betrachtet. Für weitere Informationen zu diesem `PSRepository` verwenden Sie den Befehl `Get-PSRepository -Name PSGallery`.

Standardmäßig werden über diesen Befehl Module im `%USERPROFILE%\Documents\WindowsPowerShell\Modules`-Ordner installiert. Um die AWS Tools for PowerShell für alle Benutzer eines Computers zu installieren, müssen Sie den folgenden Befehl in einer PowerShell

Sitzung ausführen, die Sie als Administrator gestartet haben. Mit dem folgenden Befehl wird beispielsweise das IAM-Modul im `%ProgramFiles%\WindowsPowerShell\Modules`-Ordner installiert, auf den alle Benutzer zugreifen können.

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

Um andere Module zu installieren, führen Sie ähnliche Befehle mit den entsprechenden Modulnamen aus, wie in der PowerShell -Galerie zu finden.

Installieren von AWSPowerShell.NetCore on Windows

Sie können die AWSPowerShell.NetCore on-Computer installieren, auf denen Windows mit PowerShell Version 3 bis 5.1 oder PowerShell Core 6.0 oder höher ausgeführt wird. Informationen zur Installation von PowerShell Core finden Sie unter Installieren verschiedener Versionen von PowerShell auf der Microsoft- PowerShell Website.

Sie können auf NetCore zwei Arten installieren AWSPowerShell.

- Laden Sie das Modul von AWSPowerShellNetCoreZIP herunter und extrahieren Sie es in einem der Modulverzeichnisse. Sie können Ihre Modulverzeichnisse erkennen, indem Sie den Wert der `PSModulePath`-Umgebungsvariablen anzeigen.
- Installieren von aus der - PowerShell Galerie mithilfe des `Install-Module` -Cmdlets, wie im folgenden Verfahren beschrieben.

So installieren Sie AWSPowerShell.NetCore from the PowerShell Gallery mit dem Cmdlet Install-Module

Um die aus NetCore der AWSPowerShell- PowerShell Galerie zu installieren, muss auf Ihrem Computer PowerShell 5.0 oder höher oder PowerShellGet PowerShell 3 oder höher ausgeführt werden. Führen Sie den folgenden Befehl aus.

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

Wenn Sie PowerShell als Administrator ausführen, wird der vorherige Befehl AWS Tools for PowerShell für alle Benutzer auf dem Computer installiert. Wenn Sie PowerShell als Standardbenutzer ohne Administratorberechtigungen ausführen, wird derselbe Befehl nur AWS Tools for PowerShell für den aktuellen Benutzer installiert.

Um nur für den aktuellen Benutzer zu installieren, wenn dieser Benutzer über Administratorberechtigungen verfügt, führen Sie den Befehl mit dem festgelegten Parameter `-Scope CurrentUser` wie folgt aus.

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

Obwohl PowerShell 3.0 und höhere Versionen Module normalerweise in Ihre PowerShell Sitzung laden, wenn Sie zum ersten Mal ein Cmdlet im Modul ausführen, ist das AWSPowerShell.NetCore module zu groß, um diese Funktionalität zu unterstützen. Sie müssen stattdessen das AWSPowerShell.NetCore Core-Modul explizit in Ihre PowerShell Sitzung laden, indem Sie den folgenden Befehl ausführen.

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

Um das AWSPowerShell.NetCore module automatisch in eine PowerShell Sitzung zu laden, fügen Sie diesen Befehl Ihrem PowerShell Profil hinzu. Weitere Informationen zum Bearbeiten Ihres PowerShell Profils finden Sie unter About Profiles in der - PowerShell Dokumentation.

Installieren von AWSPowerShell unter Windows PowerShell

Sie können die AWS Tools for Windows PowerShell auf zwei Arten installieren:

- Laden Sie das Modul von AWSPowerShellZIP herunter und extrahieren Sie es in einem der Modulverzeichnisse. Sie können Ihre Modulverzeichnisse erkennen, indem Sie den Wert der `PSModulePath`-Umgebungsvariablen anzeigen.
- Installieren von aus der - PowerShell Galerie mithilfe des `Install-Module` -Cmdlets, wie im folgenden Verfahren beschrieben.

So installieren Sie AWSPowerShell aus der - PowerShell Galerie mit dem Cmdlet Install-Module

Sie können die AWSPowerShell aus der - PowerShell Galerie installieren, wenn Sie PowerShell 5.0 oder höher ausführen oder PowerShellGet auf PowerShell 3 oder höher installiert haben. Sie können AWSPowerShell über Microsoft Gallery installieren und aktualisierenPowerShell, indem Sie den folgenden Befehl ausführen.

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

Um das AWSPowerShell Modul automatisch in eine PowerShell Sitzung zu laden, fügen Sie Ihrem PowerShell Profil das vorherige `import-module` Cmdlet hinzu. Weitere Informationen zum Bearbeiten Ihres PowerShell Profils finden Sie unter About Profiles in der - PowerShell Dokumentation.

> **Note**
>
> Die Tools für Windows PowerShell sind standardmäßig auf allen Windows-basierten Amazon Machine Images (AMIs) installiert.

Aktivieren der Skriptausführung

Um die AWS Tools for PowerShell Module zu laden, müssen Sie die PowerShell Skriptausführung aktivieren. Weisen Sie zur Aktivierung der Skriptausführung mit dem Cmdlet `Set-ExecutionPolicy` die Richtlinie `RemoteSigned` zu. Weitere Informationen finden Sie auf der Microsoft-Technet-Website unter About Execution Policies (Über Ausführungsrichtlinien).

> **Note**
>
> Dies ist nur für Computer erforderlich, auf denen Windows ausgeführt wird. Die Sicherheitseinschränkung `ExecutionPolicy` ist auf anderen Betriebssystemen nicht vorhanden.

So aktivieren Sie die Skriptausführung

1. Zum Festlegen der Ausführungsrichtlinien sind Administratorrechte erforderlich. Wenn Sie nicht als Benutzer mit Administratorrechten angemeldet sind, öffnen Sie eine PowerShell Sitzung als Administrator. Wählen Sie Start und dann All Programs (Alle Programme). Wählen Sie Telefonie und dann Windows aus PowerShell. Klicken Sie mit der rechten Maustaste auf Windows PowerShell und wählen Sie im Kontextmenü Als Administrator ausführen aus.
2. Geben Sie in der Eingabeaufforderung Folgendes ein.

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

> **Note**
>
> Auf einem 64-Bit-System müssen Sie dies für die 32-Bit-Version von Windows PowerShell PowerShell (x86) separat tun.

Wenn Sie die Ausführungsrichtlinie nicht korrekt festgelegt haben, PowerShell zeigt den folgenden Fehler an, wenn Sie versuchen, ein Skript auszuführen, z. B. Ihr 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
```

Das PowerShell Installationsprogramm von Tools for Windows aktualisiert den PSModulePath automatisch, sodass er den Speicherort des Verzeichnisses enthält, das das `AWSPowerShell` Modul enthält.

Da die den Speicherort des AWS Modulverzeichnisses `PSModulePath` enthält, zeigt das `Get-Module -ListAvailable` Cmdlet das Modul an.

```
PS > Get-Module -ListAvailable

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

Versionsverwaltung

AWS veröffentlicht AWS Tools for PowerShell regelmäßig neue Versionen von , um neue AWS Services und Funktionen zu unterstützen. Um die Version der Tools zu ermitteln, die Sie installiert haben, führen Sie das Get-cmdletAWSPowerShellVersion aus.

```
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]
```

Sie können den `-ListServiceVersionInfo` Parameter auch einem Get-AWSPowerShellVersionBefehl hinzufügen, um eine Liste der AWS Services anzuzeigen, die in der aktuellen Version der Tools unterstützt werden. Wenn Sie die modularisierte Option `AWS.Tools.*` verwenden, werden nur die Module angezeigt, die Sie aktuell importiert haben.

```
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
...
```

Um die Version von zu ermitteln PowerShell , die Sie ausführen, geben Sie ein, `$PSVersionTable` um den Inhalt der VersionTable automatischen $PS-Variable anzuzeigen.

```
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
```

Aktualisieren der AWS Tools for PowerShell unter Windows

Wenn aktualisierte Versionen von veröffentlicht AWS Tools for PowerShell werden, sollten Sie regelmäßig die Version aktualisieren, die Sie lokal ausführen.

Aktualisieren der modularisierten **`AWS.Tools`** Module

Führen Sie den folgenden Befehl aus, um Ihre `AWS.Tools` Module auf die neueste Version zu aktualisieren:

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

Mit diesem Befehl werden alle aktuell installierten `AWS.Tools`-Module aktualisiert und nach erfolgreicher Aktualisierung andere installierte Versionen entfernt.

> **Note**
>
> Über das Cmdlet `Update-AWSToolsModule` werden alle Module aus dem `PSRepository` namens `PSGallery` heruntergeladen (https://www.powershellgallery.com/) und als vertrauenswürdige Quelle betrachtet. Für weitere Informationen zu diesem `PSRepository` verwenden Sie den Befehl `Get-PSRepository -Name PSGallery`.

Aktualisieren der Tools for PowerShell Core

Führen Sie das `Get-AWSPowerShellVersion` Cmdlet aus, um die Version zu ermitteln, die Sie ausführen, und vergleichen Sie diese mit der Version von Tools for Windows PowerShell , die auf der PowerShell Gallery-Website verfügbar ist. Wir empfehlen Ihnen, dies alle zwei bis drei Wochen zu überprüfen. Unterstützung für neue Befehle und AWS Services ist erst verfügbar, nachdem Sie auf eine Version mit dieser Unterstützung aktualisiert haben.

Bevor Sie eine neuere Version von installieren AWSPowerShellNetCore, deinstallieren Sie das vorhandene Modul. Schließen Sie alle offenen PowerShell Sitzungen, bevor Sie das vorhandene Paket deinstallieren. Führen Sie zur Deinstallation den folgenden Befehl aus.

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

Nachdem das Paket deinstalliert wurde, installieren Sie das aktualisierte Modul, indem Sie den folgenden Befehl ausführen.

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

Führen Sie nach der Installation den Befehl aus, `Import-Module AWSPowerShell.NetCore` um die aktualisierten Cmdlets in Ihre PowerShell Sitzung zu laden.

Aktualisieren der Tools für Windows PowerShell

Führen Sie das `Get-AWSPowerShellVersion` Cmdlet aus, um die Version zu ermitteln, die Sie ausführen, und vergleichen Sie diese mit der Version von Tools for Windows PowerShell , die auf der PowerShell Gallery-Website verfügbar ist. Wir empfehlen Ihnen, dies alle zwei bis drei Wochen zu überprüfen. Unterstützung für neue Befehle und AWS Services ist erst verfügbar, nachdem Sie auf eine Version mit dieser Unterstützung aktualisiert haben.

- Wenn Sie die Installation mithilfe des Cmdlets `Install-Module` vorgenommen haben, führen Sie die folgenden Befehle aus.

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

- Wenn Sie die Installation mithilfe einer heruntergeladenen ZIP-Datei durchgeführt haben:
  1. Laden Sie die neueste Version von der Website Tools for PowerShell herunter. Vergleichen Sie die Paketversionsnummer im heruntergeladenn Dateinamen mit der Versionsnummer, die Sie beim Ausführen des Cmdlets `Get-AWSPowerShellVersion` erhalten.
  2. Wenn die Download-Version höher ist als die von Ihnen installierte Version, schließen Sie alle PowerShell Konsolen von Tools for Windows.
  3. Installieren Sie die neuere Version der Tools for Windows PowerShell.

Führen Sie nach der Installation aus, `Import-Module AWSPowerShell` um die aktualisierten Cmdlets in Ihre PowerShell Sitzung zu laden. Oder führen Sie die benutzerdefinierte AWS Tools for PowerShell Konsole über das Start menüaus.

Installation AWS Tools for PowerShell unter Linux oder macOS

Dieses Thema enthält Anweisungen zur Installation von AWS Tools for PowerShell unter Linux oder macOS.

Übersicht über die Einrichtung

Um AWS Tools for PowerShell auf einem Linux- oder macOS-Computer zu installieren, können Sie aus zwei Paketoptionen wählen:

- **`AWS.Tools`** – Die modularisierte Version von AWS Tools for PowerShell. Jeder AWS Service wird von einem eigenen, kleinen Modul mit gemeinsam genutzten Supportmodulen unterstütztAWS.Tools.Common.
- AWSPowerShell.NetCore – Die einzelne Large-Modul-Version von AWS Tools for PowerShell. Alle - AWS Services werden von diesem einzigen, großen Modul unterstützt.

> **Note**
>
> Beachten Sie, dass das einzelne Modul möglicherweise zu groß ist, um es mit AWS Lambda-Features zu verwenden. Verwenden Sie stattdessen die oben gezeigte modularisierte Version.

Das Einrichten eines dieser Paket auf einem Computer unter Linux oder macOS umfasst die folgenden Aufgaben, die weiter unten in diesem Thema ausführlich beschrieben werden:

1. Installieren Sie PowerShell Core 6.0 oder höher auf einem unterstützten System.
2. Nachdem Sie PowerShell Core installiert haben, führen Sie PowerShell zunächst `pwsh` in Ihrer System-Shell aus.
3. Installieren Sie entweder `AWS.Tools` oder AWSPowerShellNetCore.
4. Führen Sie das entsprechende `Import-Module` Cmdlet aus, um das Modul in Ihre PowerShell Sitzung zu importieren.
5. Führen Sie das Initialize-AWSDefaultConfigurationcmdlet aus, um Ihre AWS Anmeldeinformationen bereitzustellen.

Voraussetzungen

Um die auszuführen AWS Tools for PowerShell Core, muss auf Ihrem Computer PowerShell Core 6.0 oder höher ausgeführt werden.

- Eine Liste der unterstützten Linux-Plattformversionen und Informationen zum Installieren der neuesten Version von PowerShell auf einem Linux-basierten Computer finden Sie unter Installieren von PowerShell unter Linux auf der Microsoft-Website. Einige Linux-basierte Betriebssysteme, wie z. B. Arch, Kali und Raspbian werden nicht offiziell unterstützt, verfügen jedoch über Community-Support unterschiedlicher Stufen.

- Informationen zu unterstützten macOS-Versionen und zur Installation der neuesten Version von PowerShell unter macOS finden Sie unter Installieren von PowerShell unter macOS auf der Microsoft-Website.

Installieren der **AWS.Tools** unter Linux oder macOS

Sie können die modularisierte Version von AWS Tools for PowerShell auf Computern installieren, auf denen Core 6.0 oder höher ausgeführt PowerShell wird. Informationen zur Installation von PowerShell Core finden Sie unter Installieren verschiedener Versionen von PowerShell auf der Microsoft- PowerShell Website.

Sie können `AWS.Tools` auf drei Arten installieren:

- Verwenden der Cmdlets im `AWS.Tools.Installer`-Modul. Dieses Modul vereinfacht die Installation und Aktualisierung anderer `AWS.Tools` Module. `AWS.Tools.Installer` erfordert `PowerShellGet`und lädt automatisch eine aktualisierte Version davon herunter und installiert sie. hält Ihre Modulversionen `AWS.Tools.Installer` automatisch synchron. Wenn Sie eine neuere Version eines Moduls installieren oder aktualisieren, aktualisieren die Cmdlets in `AWS.Tools.Installer` automatisch alle Ihre anderen `AWS.Tools` Module auf dieselbe Version.

  Diese Methode wird im folgenden Verfahren beschrieben.
- Laden Sie die Module von `AWS.Tools.zip` herunter und extrahieren Sie sie in einem der Modulverzeichnisse. Sie können Ihre Modulverzeichnisse erkennen, indem Sie den Wert der `$Env:PSModulePath`-Variablen drucken.
- Installieren jedes Servicemoduls aus der - PowerShell Galerie mithilfe des `Install-Module` - Cmdlets.

So installieren Sie **AWS.Tools** unter Linux oder macOS mit dem -**AWS.Tools.Installer**Modul

1. Starten Sie eine PowerShell Core-Sitzung, indem Sie den folgenden Befehl ausführen.

   ```
   $ pwsh
   ```

   > **Note**
   >
   > Wir empfehlen, nicht PowerShell als Administrator mit erhöhten Berechtigungen auszuführen, es sei denn, dies ist für die jeweilige Aufgabe erforderlich. Grund ist das

potenzielle Sicherheitsrisiko und weil dies im Widerspruch zum Prinzip der geringsten Zugriffsrechte stünde.

2. Führen Sie den folgenden Befehl aus, um das modularisierte `AWS.Tools`-Paket mit dem `AWS.Tools.Installer`-Modul zu installieren.

```
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
```

Wenn Sie benachrichtigt werden, dass das Repository "nicht vertrauenswürdig" ist, werden Sie gefragt, ob trotzdem installiert werden soll. Geben Sie ein**y**, um die Installation des Moduls PowerShell zu erlauben. Um die Eingabeaufforderung zu vermeiden und das Modul zu installieren, ohne dem Repository zu vertrauen, können Sie den folgenden Befehl ausführen:

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

3. Sie können das Modul jetzt für jeden Dienst installieren, den Sie verwenden möchten. Mit dem folgenden Befehl werden beispielsweise die Module Amazon EC2 und Amazon S3 installiert. Mit diesem Befehl werden auch alle abhängigen Module installiert, die für die Funktionsfähigkeit des angegebenen Moduls erforderlich sind. Wenn Sie beispielsweise Ihr erstes `AWS.Tools`-Servicemodul installieren, wird damit auch `AWS.Tools.Common` installiert. Dies ist ein gemeinsam genutztes Modul, das von allen AWS Servicemodulen benötigt wird. Damit werden auch ältere Versionen der Module entfernt und weitere Module auf dieselbe neuere Version aktualisiert.

```
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**
>
> Über das Cmdlet `Install-AWSToolsModule` werden alle Module aus dem `PSRepository` namens `PSGallery` heruntergeladen (https://www.powershellgallery.com/) und das Repository als vertrauenswürdige Quelle betrachtet. Für weitere Informationen zu diesem `PSRepository` verwenden Sie den Befehl `Get-PSRepository -Name PSGallery`.

Mit dem vorherigen Befehl werden Module in den Standardverzeichnissen auf Ihrem System installiert. Die tatsächlichen Verzeichnisse hängen von Ihrer Betriebssystemverteilung und -version sowie von der PowerShell installierten Version von ab. Wenn Sie beispielsweise PowerShell 7 auf einem RHEL-ähnlichen System installiert haben, befinden sich die Standardmodule höchstwahrscheinlich in `/opt/microsoft/powershell/7/Modules` (oder `$PSHOME/Modules`) und Benutzermodule höchstwahrscheinlich in `~/.local/share/powershell/Modules`. Weitere Informationen finden Sie unter Installieren PowerShell von unter Linux auf der Microsoft- PowerShell Website. Wenn Sie anzeigen möchten, wo Module installiert sind, führen Sie den folgenden Befehl aus:

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

Um andere Module zu installieren, führen Sie ähnliche Befehle mit den entsprechenden Modulnamen aus, wie in der PowerShell -Galerie zu finden.

Installieren von AWSPowerShell.NetCore on Linux oder macOS

Um auf eine neuere Version von zu aktualisieren AWSPowerShellNetCore, folgen Sie den Anweisungen unter Aktualisieren der AWS Tools for PowerShell unter Linux oder macOS. Deinstallieren Sie frühere Versionen von AWSPowerShell.NetCore first.

Sie können auf AWSPowerShellNetCore zwei Arten installieren:

- Laden Sie das Modul von `AWSPowerShell.NetCore.zip` herunter und extrahieren Sie es in einem der Modulverzeichnisse. Sie können Ihre Modulverzeichnisse erkennen, indem Sie den Wert der `$Env:PSModulePath`-Variablen drucken.
- Installieren von aus der - PowerShell Galerie mithilfe des `Install-Module` -Cmdlets, wie im folgenden Verfahren beschrieben.

So installieren Sie AWSPowerShell.NetCore on Linux oder macOS mit dem Cmdlet Install-Module

Starten Sie eine PowerShell Core-Sitzung, indem Sie den folgenden Befehl ausführen.

```
$ pwsh
```

> **Note**
>
> Wir empfehlen Ihnen, nicht mit der Ausführung PowerShell von PowerShell mit erhöhten Administratorrechten `sudo pwsh` zu beginnen. Grund ist das potenzielle Sicherheitsrisiko und weil dies im Widerspruch zum Prinzip der geringsten Zugriffsrechte stünde.

Führen Sie den folgenden Befehl aus, um das AWSPowerShell.NetCore single-module-Paket aus der - PowerShell Galerie zu installieren.

```
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
```

Wenn Sie benachrichtigt werden, dass das Repository "nicht vertrauenswürdig" ist, werden Sie gefragt, ob trotzdem installiert werden soll. Geben Sie ein**y**, um die Installation des Moduls PowerShell zu erlauben. Um die Eingabeaufforderung zu vermeiden, ohne dem Repository zu vertrauen, können Sie den folgenden Befehl ausführen:

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

Sie müssen diesen Befehl nicht als Stamm ausführen, es sei denn, Sie möchten die AWS Tools for PowerShell für alle Benutzer eines Computers installieren. Führen Sie dazu den folgenden Befehl in einer PowerShell Sitzung aus, die Sie mit gestartet habensudo pwsh.

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

Skriptausführung

Der Befehl `Set-ExecutionPolicy` ist auf Nicht-Windows-Systemen nicht verfügbar. Sie können ausführen`Get-ExecutionPolicy`, was zeigt, dass die Standardeinstellung für die Ausführungsrichtlinie in PowerShell Core, die auf Nicht-Windows-Systemen ausgeführt wird, ist`Unrestricted`. Weitere Informationen finden Sie auf der Microsoft-Technet-Website unter About Execution Policies (Über Ausführungsrichtlinien).

Da die den Speicherort des AWS Modulverzeichnisses `PSModulePath` enthält, zeigt das `Get-Module -ListAvailable` Cmdlet das installierte Modul an.

`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
```

Konfigurieren einer PowerShell Konsole für die Verwendung der AWS Tools for PowerShell Core (AWSPowerShellNetCore nur)

PowerShell Core lädt Module in der Regel automatisch, wenn Sie ein Cmdlet im Modul ausführen. Dies funktioniert jedoch nicht für AWSPowerShell. NetCore Aufgrund seiner großen Größe. Um mit der Ausführung von AWSPowerShell.NetCore cmdlets zu beginnen, müssen Sie zuerst den `Import-Module AWSPowerShell.NetCore` Befehl ausführen. Dies ist für Cmdlets in `AWS.Tools`-Modulen nicht erforderlich.

Initialisieren Ihrer PowerShell Sitzung

Wenn Sie nach der Installation der PowerShell auf einem Linux-basierten oder macOS-basierten System beginnen AWS Tools for PowerShell, müssen Sie Initialize-AWSDefaultConfiguration ausführen, um anzugeben, welcher AWS Zugriffsschlüssel verwendet werden soll. Mehr über `Initialize-AWSDefaultConfiguration` erfahren Sie unter Verwenden von AWS-Anmeldeinformationen.

> **Note**
>
> In früheren Versionen (vor 3.3.96.0) von wurde AWS Tools for PowerShell dieses Cmdlet als bezeichnet`Initialize-AWSDefaults`.

Versionsverwaltung

AWS veröffentlicht AWS Tools for PowerShell regelmäßig neue Versionen von , um neue AWS Services und Funktionen zu unterstützen. Um die Version des zu ermitteln AWS Tools for PowerShell , das Sie installiert haben, führen Sie das Get-cmdletAWSPowerShellVersion aus.

```
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]
```

Um eine Liste der unterstützten AWS Services in der aktuellen Version der Tools anzuzeigen, fügen Sie den `-ListServiceVersionInfo` Parameter zu einem Get-AWSPowerShellVersion-Cmdlet hinzu.

Um die Version von zu ermitteln, PowerShell die Sie ausführen, geben Sie ein, `$PSVersionTable` um den Inhalt der `$PSVersionTable` automatischen Variablen anzuzeigen.

```
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
```

Aktualisieren der AWS Tools for PowerShell unter Linux oder macOS

Wenn aktualisierte Versionen von veröffentlicht AWS Tools for PowerShell werden, sollten Sie regelmäßig die Version aktualisieren, die Sie lokal ausführen.

Aktualisieren der modularisierten **`AWS.Tools`** Module

Führen Sie den folgenden Befehl aus, um Ihre `AWS.Tools` Module auf die neueste Version zu aktualisieren:

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

Mit diesem Befehl werden alle aktuell installierten `AWS.Tools`-Module aktualisiert und die früheren Versionen für die Module, die erfolgreich aktualisiert wurden, entfernt.

> **Note**
>
> Über das Cmdlet `Update-AWSToolsModule` werden alle Module aus dem `PSRepository` namens `PSGallery` heruntergeladen (https://www.powershellgallery.com/) und als vertrauenswürdige Quelle betrachtet. Für weitere Informationen zu diesem `PSRepository` verwenden Sie den Befehl `Get-PSRepository -Name PSGallery`.

Aktualisieren der Tools for PowerShell Core

Führen Sie das `Get-AWSPowerShellVersion` Cmdlet aus, um die Version zu ermitteln, die Sie ausführen, und vergleichen Sie diese mit der Version von Tools for Windows PowerShell , die auf der PowerShell Gallery-Website verfügbar ist. Wir empfehlen Ihnen, dies alle zwei bis drei Wochen zu überprüfen. Die Unterstützung für neue Befehle und AWS Services ist erst verfügbar, nachdem Sie auf eine Version mit dieser Unterstützung aktualisiert haben.

Bevor Sie eine neuere Version von installieren AWSPowerShellNetCore, deinstallieren Sie das vorhandene -Modul. Schließen Sie alle offenen PowerShell Sitzungen, bevor Sie das vorhandene Paket deinstallieren. Führen Sie zur Deinstallation den folgenden Befehl aus.

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

Nachdem das Paket deinstalliert wurde, installieren Sie das aktualisierte Modul, indem Sie den folgenden Befehl ausführen.

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

Führen Sie nach der Installation den Befehl aus, `Import-Module AWSPowerShell.NetCore` um die aktualisierten Cmdlets in Ihre PowerShell Sitzung zu laden.

Verwandte Informationen

- Erste Schritte mit der AWS Tools for Windows PowerShell

- Arbeiten mit AWS-Services in AWS Tools for PowerShell

Migrieren von der AWS Tools for PowerShell-Version 3.3 zu Version 4

AWS Tools for PowerShell Version 4 ist ein abwärtskompatibles Update auf AWS Tools for PowerShell Version 3.3. Es wurde um erhebliche Verbesserungen ergänzt, gleichzeitig bleibt das vorhandene Cmdlet-Verhalten beibehalten.

Ihre vorhandenen Skripts sollten nach dem Upgrade auf die neue Version weiterhin funktionieren. Wir empfehlen jedoch, sie gründlich zu testen, bevor Sie Ihre Produktionsumgebungen aktualisieren.

In diesem Abschnitt werden die Änderungen beschrieben und wird erläutert, wie sich diese auf Ihre Skripts auswirken können.

Vollständig modularisierte **`AWS.Tools`**-Version

Die Pakete AWSPowerShell. NetCore und AWSPowerShell waren „monolithisch“. Dies bedeutete, dass alle AWS-Services im selben Modul unterstützt wurden, wodurch es sehr groß wurde und mit jedem neuen AWS-Service und jeder neuen Funktion noch größer wurde. Das neue `AWS.Tools` Paket ist in kleinere Module unterteilt, die Ihnen die Flexibilität geben, nur diejenigen herunterzuladen und zu installieren, die Sie für die von Ihnen verwendeten AWS-Services benötigen. Das Paket enthält ein gemeinsames `AWS.Tools.Common`-Modul, das von allen anderen Modulen benötigt wird, und ein `AWS.Tools.Installer`-Modul, das das Installieren, Aktualisieren und Entfernen von Modulen bedarfsgerecht vereinfacht.

Dies ermöglicht auch das automatische Importieren von Cmdlets beim ersten Aufruf, ohne dass zunächst `Import-module` aufgerufen werden muss. Um jedoch vor dem Aufruf eines Cmdlets mit den zugehörigen .NET-Objekten zu interagieren, müssen Sie trotzdem aufrufen, `Import-Module` um PowerShell über die relevanten .NET-Typen informiert zu werden.

Der folgende Befehl umfasst beispielsweise einen Verweis auf `Amazon.EC2.Model.Filter`. Dieser Referenztyp kann den automatischen Import nicht auslösen. Damit der Befehl nicht fehlschlägt, müssen Sie zunächst `Import-Module` aufrufen.

```
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
```

Neues **`Get-AWSService`**-Cmdlet

Damit Sie die Namen der Module für jeden AWS-Service in der `AWS.Tools` Modulsammlung ermitteln können, können Sie das `Get-AWSService`-Cmdlet verwenden.

```
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
  ...
```

Neuer **`-Select`**-Parameter zum Steuern des von einem Cmdlet zurückgegebenen Objekts

Die meisten Cmdlets der Version 4 unterstützen einen neuen `-Select`-Parameter. Jedes Cmdlet ruft für Sie mittels AWS SDK for .NET die AWS-Service-APIs auf. Anschließend konvertiert der AWS Tools for PowerShell Client die Antwort in ein Objekt, das Sie in Ihren PowerShell Skripten verwenden können, und Pipe-Befehlen an andere Befehle. Manchmal enthält das endgültige PowerShell Objekt mehr Felder oder Eigenschaften in der ursprünglichen Antwort, als Sie benötigen, und andere Male möchten Sie möglicherweise, dass das Objekt Felder oder Eigenschaften der Antwort enthält, die standardmäßig nicht vorhanden sind. Mit dem `-Select`-Parameter können Sie angeben, was in dem vom Cmdlet zurückgegebenen NET-Objekt enthalten sein soll.

Beispielsweise ruft das Get-S3Object-Cmdlet die Amazon S3-SDK-Operation aufListObjects. Diese Operation gibt ein -ListObjectsResponseObjekt zurück. Standardmäßig gibt das `Get-S3Object`

Cmdlet jedoch nur das `S3Objects` Element der SDK-Antwort an den PowerShell Benutzer zurück. Im folgenden Beispiel ist dieses Objekt ein Array mit zwei Elementen.

```
PS > Get-S3Object -BucketName mybucket

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

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

In AWS Tools for PowerShell Version 4 können Sie `-Select *` so definieren, dass das vollständige .NET-Antwortobjekt zurückgegeben wird, das vom SDK-API-Aufruf zurückgegeben wird.

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

Sie können auch den Pfad zu der gewünschten verschachtelten Eigenschaft angeben. Im folgenden Beispiel wird nur die `Key`-Eigenschaft jedes Elements im `S3Objects`-Array zurückgegeben.

```
PS > Get-S3Object -BucketName mybucket -Select S3Objects.Key
file1.txt
file2.txt
```

In bestimmten Situationen kann es sinnvoll sein, einen Cmdlet-Parameter zurückzugeben. Sie können dies mit `-Select ^ParameterName` tun. Diese Funktion ersetzt den `-PassThru`-Parameter, der zwar noch verfügbar, aber veraltet ist.

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

Das Referenzthema für jedes Cmdlet erkennt, ob es den Parameter `-Select` unterstützt.

Konsistentere Begrenzung der Anzahl der Elemente in der Ausgabe

In früheren Versionen von AWS Tools for PowerShell konnten Sie mit dem `-MaxItems`-Parameter die maximale Anzahl von Objekten angeben, die in der endgültigen Ausgabe zurückgegeben werden sollen.

Dieses Verhalten wird aus `AWS.Tools` entfernt.

Dieses Verhalten ist in und veraltet AWSPowerShellNetCore AWSPowerShellund wird in einer zukünftigen Version aus diesen Versionen entfernt.

Wenn die zugrunde liegende Service-API einen `MaxItems`-Parameter unterstützt, ist er weiterhin verfügbar und funktioniert wie in der API angegeben. Aber er verfügt nicht mehr über das zusätzliche Verhalten, die Anzahl der Elemente zu begrenzen, die in der Ausgabe des Cmdlets zurückgegeben werden.

Um die Anzahl der Elemente zu begrenzen, die in der endgültigen Ausgabe zurückgegeben werden, übergeben Sie die Ausgabe an das `Select-Object`-Cmdlet und geben Sie den `-First` *n*-Parameter an, wobei *n* die maximale Anzahl von Elementen ist, die in die endgültige Ausgabe aufgenommen werden sollen.

```
PS > Get-S3ObjectV2 -BucketName BUCKET_NAME -Select S3Objects.Key | select -first 2
file1.txt
file2.txt
```

Nicht alle AWS-Services unterstützten `-MaxItems` auf die gleiche Weise, sodass diese Inkonsistenz und die unerwarteten Ergebnisse, die manchmal vorgekommen sind, damit beseitigt werden.

Außerdem konnte `-MaxItems` in Kombination mit dem neuen `-Select`-Parameter zuweilen zu irritierenden Ergebnissen führen.

Einfachere Verwendung von Stream-Parametern

Parameter vom Typ `Stream` oder `byte[]` können nun `string`-, `string[]`- oder `FileInfo`-Werte übernehmen.

Sie können z. B. eines der folgenden Beispiele verwenden.

```
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 wandelt mittels UTF-8-Codierung alle Zeichenfolgen in `byte[]` um.

Erweitern der Pipe um den Eigenschaftsnamen

Um die Benutzererfahrung konsistenter zu gestalten, können Sie nun Pipeline-Eingaben übergeben, indem Sie den Eigenschaftsnamen für einen beliebigen Parameter angeben.

Im folgenden Beispiel erstellen wir ein benutzerdefiniertes Objekt mit Eigenschaften, die Namen aufweisen, die mit den Parameternamen des Ziel-Cmdlets übereinstimmen. Wenn das Cmdlet ausgeführt wird, werden diese Eigenschaften automatisch als dessen Parameter verwendet.

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

> **Note**
>
> Einige Eigenschaften unterstützten dies in früheren Versionen von AWS Tools for PowerShell. Mit Version 4 erfolgt dies konsequenter, indem es für alle Parameter aktiviert wird.

Statische gängige Parameter

Um die Konsistenz in Version 4.0 von AWS Tools for PowerShell zu verbessern, sind alle Parameter statisch.

In früheren Versionen von AWS Tools for PowerShell waren einige gängige Parameter wie `AccessKey`,`SecretKey`, `ProfileName` oder `Region` dynamisch, während alle anderen Parameter statisch waren. Dies könnte zu Problemen führen, da statische Parameter vor dynamischen Parametern PowerShell bindet. Angenommen, Sie haben den folgenden Befehl ausgeführt:

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

Frühere Versionen von PowerShell binden den Wert `us-west-2` an den `-RegionName` statischen Parameter anstelle des `-Region` dynamischen Parameters an. Dies konnte Benutzer durchaus irritieren.

`AWS.Tools` deklariert und erzwingt obligatorische Parameter

Die Module `AWS.Tools.*` deklarieren und erzwingen nun obligatorische Cmdlet-Parameter. Wenn ein -AWSService angibt, dass ein Parameter einer API erforderlich ist, PowerShell fordert Sie auf, den entsprechenden Cmdlet-Parameter einzugeben, wenn Sie ihn nicht angegeben haben. Dies gilt nur für `AWS.Tools`. Um die Abwärtskompatibilität zu gewährleisten, gilt dies nicht für AWSPowerShell. NetCore oder AWSPowerShell.

Alle Parameter sind löschbar

Sie können nun den Werttyp-Parametern (Zahlen und Daten) `$null` zuordnen. Diese Änderung sollte sich nicht auf vorhandene Skripte auswirken. Auf diese Weise können Sie die Eingabeaufforderung für einen obligatorischen Parameter umgehen. Obligatorische Parameter werden nur in `AWS.Tools` erzwungen.

Wenn Sie das folgende Beispiel mit Version 4 ausführen, wird die clientseitige Validierung effektiv umgangen, da Sie für jeden obligatorischen Parameter einen „Wert“ angeben. Der Aufruf des Amazon-EC2-API-Services schlägt jedoch fehl, da der AWS-Service diese Information weiterhin benötigt.

```
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
```

Entfernen von zuvor nicht unterstützten Funktionen

Die folgenden Funktionen wurden in früheren Releases von AWS Tools for PowerShell nicht unterstützt und werden in Version 4 entfernt:

- Der `-Terminate`-Parameter wurde aus dem `Stop-EC2Instance`-Cmdlet entfernt. Verwenden Sie stattdessen `Remove-EC2Instance`.
- Der `-ProfileName` Parameter wurde aus dem Clear-AWSCredential cmdlet entfernt. Verwenden Sie stattdessen `Remove-AWSCredentialProfile`.
- Es wurden die Cmdlets `Import-EC2Instance` und `Import-EC2Volume` entfernt.

Erste Schritte mit der AWS Tools for Windows PowerShell

Einige der Themen in diesem Abschnitt beschreiben die Grundlagen der Verwendung der Tools für Windows PowerShell nach der Installation der Tools. Sie erfahren z. B., wie Sie die Anmeldeinformationen und die AWS-Region festlegen, die die Tools for Windows PowerShell bei der Interaktion mit AWS verwenden sollen.

Andere Themen in diesem Abschnitt enthalten Informationen zu erweiterten Möglichkeiten für die Konfiguration der Tools, Ihrer Umgebung und Ihrer Projekte.

Themen

Konfigurieren der Tool-Authentifizierung mit AWS

Sie müssen bei der Entwicklung mit festlegen AWS , wie sich Ihr Code bei authentifiziert AWS-Services. Es gibt verschiedene Möglichkeiten, den programmgesteuerten Zugriff auf - AWS Ressourcen zu konfigurieren, je nach Umgebung und verfügbarem AWS Zugriff.

Informationen zu verschiedenen Authentifizierungsmethoden für die Tools für PowerShellfinden Sie unter Authentifizierung und Zugriff im AWS Referenzhandbuch für -SDKs und Tools.

In diesem Thema wird davon ausgegangen, dass ein neuer Benutzer lokal entwickelt, von seinem Arbeitgeber keine Authentifizierungsmethode erhalten hat und verwendet, AWS IAM Identity Center um temporäre Anmeldeinformationen zu erhalten. Wenn Ihre Umgebung nicht unter diese Annahmen fällt, treffen einige der Informationen in diesem Thema möglicherweise nicht auf Sie zu, oder einige der Informationen wurden Ihnen möglicherweise bereits gegeben.

Die Konfiguration dieser Umgebung erfordert mehrere Schritte, die sich wie folgt zusammenfassen lassen:

1. Aktivieren und Konfigurieren von IAM Identity Center
2. Konfigurieren Sie die Tools für PowerShell zur Verwendung von IAM Identity Center.
3. Starten einer - AWS Zugriffsportalsitzung

Aktivieren und Konfigurieren von IAM Identity Center

Um verwenden zu können AWS IAM Identity Center, muss es zuerst aktiviert und konfiguriert werden. Weitere Informationen dazu, wie Sie dies für tun PowerShell, finden Sie in Schritt 1 im Thema zur IAM-Identity-Center-Authentifizierung im AWS Referenzhandbuch zu -SDKs und Tools. Folgen Sie insbesondere den Anweisungen unter Ich habe keinen Zugriff über IAM Identity Center eingerichtet.

Konfigurieren Sie die Tools für PowerShell zur Verwendung von IAM Identity Center.

> **Note**
>
> Beginnend mit Version 4.1.538 der Tools für besteht die empfohlene Methode zum Konfigurieren von SSO-Anmeldeinformationen und zum Starten einer AWS - Zugriffsportalsitzung darin PowerShell, die `Invoke-AWSSSOLogin` Cmdlets `Initialize-AWSSSOConfiguration` und zu verwenden, wie in diesem Thema beschrieben. Wenn Sie keinen Zugriff auf diese Version der Tools für PowerShell (oder höher) haben oder diese Cmdlets nicht verwenden können, können Sie diese Aufgaben trotzdem mithilfe der ausführen AWS CLI. Informationen dazu finden Sie unter Verwenden der AWS CLI für die Portalanmeldung.

Das folgende Verfahren aktualisiert die AWS `config` freigegebene Datei mit SSO-Informationen, die die Tools für zum Abrufen temporärer Anmeldeinformationen PowerShell verwenden. Als Folge dieses Verfahrens wird auch eine AWS -Zugriffsportalsitzung gestartet. Wenn die freigegebene `config` Datei bereits SSO-Informationen enthält und Sie nur wissen möchten, wie Sie eine - Zugriffsportalsitzung mit den Tools für starten PowerShell, lesen Sie den nächsten Abschnitt in diesem Thema, Starten einer - AWS Zugriffsportalsitzung.

1. Wenn Sie dies noch nicht getan haben, öffnen PowerShell und installieren Sie die AWS Tools for PowerShell entsprechend Ihrem Betriebssystem und Ihrer Umgebung, einschließlich der gängigen Cmdlets. Weitere Informationen über die entsprechende Vorgehensweise finden Sie unter Installieren des AWS Tools for PowerShell.

   Wenn Sie beispielsweise die modularisierte Version der Tools für PowerShell unter Windows installieren, würden Sie höchstwahrscheinlich Befehle ausführen, die den folgenden ähneln:

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

2. Führen Sie den folgenden Befehl aus. Ersetzen Sie die Beispieleigenschaftswerte durch Werte aus Ihrer IAM-Identity-Center-Konfiguration. Informationen zu diesen Eigenschaften und deren Suche finden Sie unter Einstellungen für IAM-Identity-Center-Anmeldeinformationsanbieter im AWS Referenzhandbuch für SDKs und Tools.

   ```
   $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
   ```

   Alternativ können Sie einfach das Cmdlet selbst, und die Tools for verwendenInitialize-AWSSSOConfiguration, um Sie zur Eingabe der Eigenschaftswerte PowerShell aufzufordern.

   Überlegungen zu bestimmten Eigenschaftswerten:

   - Wenn Sie einfach die Anweisungen zum Aktivieren und Konfigurieren von IAM Identity Center befolgt haben, ist der Wert für `-RoleName` möglicherweise `PowerUserAccess`. Wenn Sie jedoch einen speziell für die PowerShell Arbeit festgelegten IAM-Identity-Center-Berechtigungssatz erstellt haben, verwenden Sie stattdessen diesen.
   - Stellen Sie sicher, dass Sie die verwenden AWS-Region , in der Sie IAM Identity Center konfiguriert haben.

3. Zu diesem Zeitpunkt enthält die AWS `config` freigegebene Datei ein Profil `my-sso-profile` namens mit einer Reihe von Konfigurationswerten, auf die von den Tools für verwiesen werden kann PowerShell. Den Speicherort dieser Datei finden Sie unter Speicherort der freigegebenen Dateien im Referenzhandbuch für AWS SDKs und Tools.

   Die Tools für PowerShell verwenden den SSO-Token-Anbieter des Profils, um Anmeldeinformationen zu erhalten, bevor Anfragen an gesendet werden AWS. Der `sso_role_name` Wert, bei dem es sich um eine IAM-Rolle handelt, die mit einem IAM-Identity-Center-Berechtigungssatz verbunden ist, sollte den Zugriff auf die in Ihrer Anwendung AWS-Services verwendeten ermöglichen.

   Das folgende Beispiel zeigt das Profil, das mit dem oben gezeigten Befehl erstellt wurde. Einige der Eigenschaftswerte und ihre Reihenfolge können in Ihrem tatsächlichen Profil unterschiedlich sein. Die `-sso-session`Eigenschaft des Profils bezieht sich auf den Abschnitt mit dem Namen `my-sso-session`, der Einstellungen zum Initiieren einer - AWS Zugriffsportalsitzung enthält.

   ```
   [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. Wenn Sie bereits über eine aktive - AWS Zugriffsportalsitzung verfügen, PowerShell informiert Sie die Tools für darüber, dass Sie bereits angemeldet sind.

   Wenn dies nicht der Fall ist, PowerShell versucht die Tools für automatisch, die SSO-Autorisierungsseite in Ihrem Standard-Webbrowser zu öffnen. Folgen Sie den Anweisungen in Ihrem Browser, die einen SSO-Autorisierungscode, einen Benutzernamen und ein Passwort sowie die Berechtigung für den Zugriff auf AWS IAM Identity Center Konten und Berechtigungssätze enthalten können.

   Die Tools für PowerShell informieren Sie darüber, dass die SSO-Anmeldung erfolgreich war.

Starten einer - AWS Zugriffsportalsitzung

Bevor Sie Befehle ausführen, die auf zugreifen AWS-Services, benötigen Sie eine aktive - AWS Zugriffsportalsitzung, damit die Tools für die IAM-Identity-Center-Authentifizierung verwenden PowerShell können, um Anmeldeinformationen aufzulösen. Um sich beim - AWS Zugriffsportal anzumelden, führen Sie den folgenden Befehl in aus PowerShell, wobei der Name des Profils `-ProfileName my-sso-profile` ist, das in der freigegebenen `config` Datei erstellt wurde, als Sie das Verfahren im vorherigen Abschnitt dieses Themas befolgt haben.

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

Wenn Sie bereits über eine aktive - AWS Zugriffsportalsitzung verfügen, PowerShell informiert Sie die Tools für darüber, dass Sie bereits angemeldet sind.

Wenn dies nicht der Fall ist, PowerShell versucht die Tools für automatisch, die SSO-Autorisierungsseite in Ihrem Standard-Webbrowser zu öffnen. Folgen Sie den Anweisungen in Ihrem Browser, die einen SSO-Autorisierungscode, einen Benutzernamen und ein Passwort sowie die Berechtigung für den Zugriff auf AWS IAM Identity Center Konten und Berechtigungssätze enthalten können.

Die Tools für PowerShell informieren Sie darüber, dass die SSO-Anmeldung erfolgreich war.

Um zu testen, ob Sie bereits über eine aktive Sitzung verfügen, führen Sie den folgenden Befehl aus, nachdem Sie das `AWS.Tools.SecurityToken` Modul nach Bedarf installiert oder importiert haben.

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

Die Antwort auf das `Get-STSCallerIdentity` Cmdlet meldet das IAM-Identity-Center-Konto und den in der freigegebenen `config` Datei konfigurierten Berechtigungssatz.

Beispiel

Im Folgenden finden Sie ein Beispiel für die Verwendung von IAM Identity Center mit den Tools für PowerShell. Dem Beispiel liegen folgende Annahmen zugrunde:

- Sie haben IAM Identity Center aktiviert und wie zuvor in diesem Thema beschrieben konfiguriert. Die SSO-Eigenschaften befinden sich im `my-sso-profile` Profil, das zuvor in diesem Thema konfiguriert wurde.

- Wenn Sie sich über die `Invoke-AWSSSOLogin` Cmdlets `Initialize-AWSSSOConfiguration` oder anmelden, verfügt der Benutzer mindestens über Leseberechtigungen für Amazon S3.
- Einige S3-Buckets stehen diesem Benutzer zur Ansicht zur Verfügung.

Installieren oder importieren Sie das `AWS.Tools.S3` Modul nach Bedarf und verwenden Sie dann den folgenden PowerShell Befehl, um eine Liste der S3-Buckets anzuzeigen.

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

Zusätzliche Informationen

- Weitere Optionen zur Authentifizierung für die Tools für PowerShell, z. B. die Verwendung von Profilen und Umgebungsvariablen, finden Sie im Konfigurationskapitel im AWS Referenzhandbuch für SDKs und Tools.
- Bei einigen Befehlen muss eine - AWS Region angegeben werden. Es gibt eine Reihe von Möglichkeiten, dies zu tun, darunter die `-Region` Cmdlet-Option, das `[default]` Profil und die `AWS_REGION` Umgebungsvariable. Weitere Informationen finden Sie unter Angeben von AWS Regionen in diesem Handbuch und in der AWS Region im AWS Referenzhandbuch zu -SDKs und Tools.
- Weitere Informationen zu bewährten Methoden finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.
- Informationen zum Erstellen kurzfristiger AWS Anmeldeinformationen finden Sie unter Temporäre Sicherheitsanmeldeinformationen im IAM-Benutzerhandbuch.
- Weitere Informationen zu anderen Anbietern von Anmeldeinformationen finden Sie unter Standardisierte Anbieter von Anmeldeinformationen im Referenzhandbuch für AWS SDKs und Tools.

Themen

- Verwenden der AWS CLI für die Portalanmeldung

Verwenden der AWS CLI für die Portalanmeldung

Beginnend mit Version 4.1.538 der Tools for besteht PowerShelldie empfohlene Methode zum Konfigurieren von SSO-Anmeldeinformationen und zum Starten einer AWS -Zugriffsportalsitzung darin, die `Invoke-AWSSSOLogin` - `Initialize-AWSSSOConfiguration` und -Cmdlets zu

verwenden, wie unter beschriebenKonfigurieren der Tool-Authentifizierung mit AWS. Wenn Sie keinen Zugriff auf diese Version der Tools für PowerShell (oder höher) haben oder diese Cmdlets nicht verwenden können, können Sie diese Aufgaben trotzdem mithilfe der ausführen AWS CLI.

Konfigurieren Sie die Tools für PowerShell zur Verwendung von IAM Identity Center über die AWS CLI.

Wenn Sie dies noch nicht getan haben, stellen Sie sicher, dass Sie IAM Identity Center aktivieren und konfigurieren, bevor Sie fortfahren.

Informationen zur Konfiguration der Tools für für PowerShell für die Verwendung von IAM Identity Center über die AWS CLI finden Sie in Schritt 2 im Thema zur IAM-Identity-Center-Authentifizierung im AWS Referenzhandbuch zu -SDKs und Tools. Nachdem Sie diese Konfiguration abgeschlossen haben, sollte Ihr System die folgenden Elemente enthalten:

- Die AWS CLI, mit der Sie eine AWS -Zugriffsportalsitzung starten, bevor Sie Ihre Anwendung ausführen.
- Die AWS `config` freigegebene Datei, die ein `[default]` Profil mit einer Reihe von Konfigurationswerten enthält, auf die über die Tools für verwiesen werden kann PowerShell. Den Speicherort dieser Datei finden Sie unter Speicherort der freigegebenen Dateien im Referenzhandbuch für AWS SDKs und Tools. Die Tools für PowerShell verwenden den SSO-Token-Anbieter des Profils, um Anmeldeinformationen zu erhalten, bevor Anfragen an gesendet werden AWS. Der `sso_role_name` Wert, bei dem es sich um eine IAM-Rolle handelt, die mit einem IAM-Identity-Center-Berechtigungssatz verbunden ist, sollte den Zugriff auf die in Ihrer Anwendung AWS-Services verwendeten ermöglichen.

  Die folgende `config` Beispieldatei zeigt ein `[default]` Profil, das bei einem SSO-Token-Anbieter eingerichtet wurde. Die `sso_session`-Einstellung des Profils bezieht sich auf den benannten `sso-session`-Abschnitt. Der `sso-session` Abschnitt enthält Einstellungen zum Initiieren einer AWS -Zugriffsportalsitzung.

```
[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**
>
> In Ihrer PowerShell Sitzung müssen die folgenden Module installiert und importiert sein, damit die SSO-Auflösung funktioniert:
>
> - `AWS.Tools.SSO`
> - `AWS.Tools.SSOOIDC`
>
> Wenn Sie eine ältere Version der Tools für verwenden PowerShell und diese Module nicht haben, erhalten Sie eine Fehlermeldung ähnlich der folgenden: „Assembly AWSSDK.SSOOIDC konnte nicht gefunden werden...“.

Starten einer - AWS Zugriffsportalsitzung

Bevor Sie Befehle ausführen, die auf zugreifen AWS-Services, benötigen Sie eine aktive - AWS Zugriffsportalsitzung, damit die Tools for Windows die IAM-Identity-Center-Authentifizierung verwenden PowerShell können, um Anmeldeinformationen aufzulösen. Abhängig von Ihren konfigurierten Sitzungslängen läuft Ihr Zugriff schließlich ab und bei den Tools for Windows PowerShell tritt ein Authentifizierungsfehler auf. Um sich beim - AWS Zugriffsportal anzumelden, führen Sie den folgenden Befehl in der aus AWS CLI.

```
aws sso login
```

Da Sie das `[default]` Profil verwenden, müssen Sie den Befehl nicht mit der `--profile` Option aufrufen. Wenn Ihre Konfiguration des SSO-Token-Anbieters ein benanntes Profil verwendet, lautet der Befehl `aws sso login --profile` *`named-profile`* stattdessen . Weitere Informationen zu benannten Profilen finden Sie im Abschnitt Profile im AWS Referenzhandbuch zu -SDKs und Tools.

Um zu testen, ob Sie bereits über eine aktive Sitzung verfügen, führen Sie den folgenden AWS CLI Befehl aus (mit der gleichen Berücksichtigung für das benannte Profil):

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

In der Antwort auf diesen Befehl sollten das in der freigegebenen `config`-Datei konfigurierte IAM-Identity-Center-Konto und der Berechtigungssatz angegeben werden.

> **Note**
>
> Wenn Sie bereits über eine aktive - AWS Zugriffsportalsitzung verfügen und ausführen`aws sso login`, müssen Sie keine Anmeldeinformationen angeben.
> Beim Anmeldevorgang werden Sie möglicherweise aufgefordert, den AWS CLI Zugriff auf Ihre Daten zu erlauben. Da die auf dem SDK für Python AWS CLI aufbaut, können Berechtigungsnachrichten Variationen des `botocore` Namens enthalten.

Beispiel

Im Folgenden finden Sie ein Beispiel für die Verwendung von IAM Identity Center mit den Tools für PowerShell. Dem Beispiel liegen folgende Annahmen zugrunde:

- Sie haben IAM Identity Center aktiviert und wie zuvor in diesem Thema beschrieben konfiguriert. Die SSO-Eigenschaften befinden sich im `[default]`-Profil.
- Wenn Sie sich über die AWS CLI mit anmeldenaws `sso login`, verfügt dieser Benutzer mindestens über Leseberechtigungen für Amazon S3.
- Einige S3-Buckets stehen diesem Benutzer zur Ansicht zur Verfügung.

Verwenden Sie die folgenden PowerShell Befehle, um eine Liste der S3-Buckets anzuzeigen:

```
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
```

Wie oben erwähnt, müssen Sie das `Get-S3Bucket` Cmdlet nicht mit der `-ProfileName` Option aufrufen, da Sie das `[default]` Profil verwenden. Wenn die Konfiguration Ihres SSO-Token-Anbieters ein benanntes Profil verwendet, lautet der Befehl `Get-S3Bucket -ProfileName` *`named-profile`*. Weitere Informationen zu benannten Profilen finden Sie im Abschnitt Profile im AWS Referenzhandbuch zu -SDKs und Tools.

Zusätzliche Informationen

- Weitere Optionen zur Authentifizierung für die Tools für PowerShell, z. B. die Verwendung von Profilen und Umgebungsvariablen, finden Sie im Konfigurationskapitel im AWS Referenzhandbuch für SDKs und Tools.
- Bei einigen Befehlen muss eine - AWS Region angegeben werden. Es gibt eine Reihe von Möglichkeiten, dies zu tun, darunter die `-Region` Cmdlet-Option, das `[default]` Profil und die `AWS_REGION` Umgebungsvariable. Weitere Informationen finden Sie unter Angeben von AWS Regionen in diesem Handbuch und in der AWS Region im AWS Referenzhandbuch zu -SDKs und Tools.
- Weitere Informationen zu bewährten Methoden finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.
- Informationen zum Erstellen kurzfristiger AWS Anmeldeinformationen finden Sie unter Temporäre Sicherheitsanmeldeinformationen im IAM-Benutzerhandbuch.
- Weitere Informationen zu anderen Anbietern von Anmeldeinformationen finden Sie unter Standardisierte Anbieter von Anmeldeinformationen im Referenzhandbuch für AWS SDKs und Tools.

Angeben von AWS Regionen

Es gibt zwei Möglichkeiten, die AWS Region anzugeben, die beim Ausführen von AWS Tools for PowerShell Befehlen verwendet werden soll:

- Verwenden Sie den allgemeinen Parameter `-Region` für einzelne Befehle.
- Verwenden Sie den Befehl `Set-DefaultAWSRegion`, um eine Standardregion für alle Befehle festzulegen.

Viele AWS Cmdlets schlagen fehl, wenn die Tools for Windows nicht herausfinden PowerShell können, welche Region verwendet werden soll. Ausnahmen sind Cmdlets für Amazon S3, Amazon SES und AWS Identity and Access Management, die automatisch auf einen globalen Endpunkt festgelegt sind.

So geben Sie die Region für einen einzelnen AWS Befehl an

Fügen Sie den Parameter `-Region` zu Ihrem Befehl hinzu, zum Beispiel folgendermaßen.

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

So legen Sie eine Standardregion für alle AWS CLI-Befehle in der aktuellen Sitzung fest

Geben Sie in der PowerShell Eingabeaufforderung den folgenden Befehl ein.

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

> **Note**
>
> Diese Einstellung bleibt nur für die aktuelle Sitzung bestehen. Um die Einstellung auf alle Ihre PowerShell Sitzungen anzuwenden, fügen Sie diesen Befehl Ihrem PowerShell Profil hinzu, wie Sie es für den `Import-Module` Befehl getan haben.

So zeigen Sie die aktuelle Standardregion für alle AWS CLI-Befehle an

Geben Sie in der PowerShell Eingabeaufforderung den folgenden Befehl ein.

```
PS > Get-DefaultAWSRegion

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

So löschen Sie die aktuelle Standardregion für alle AWS CLI-Befehle

Geben Sie in der PowerShell Eingabeaufforderung den folgenden Befehl ein.

```
PS > Clear-DefaultAWSRegion
```

So zeigen Sie eine Liste aller verfügbaren AWS Regionen an

Geben Sie in der PowerShell Eingabeaufforderung den folgenden Befehl ein. Die dritte Spalte in der Beispielausgabe gibt an, welche Region die Standardeinstellung für Ihre aktuelle Sitzung ist.

```
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**
>
> Einige Regionen werden möglicherweise unterstützt, sind jedoch nicht in den Ausgaben des Cmdlets `Get-AWSRegion` enthalten. Dies gilt beispielsweise manchmal für Regionen, die noch nicht global sind. Wenn Sie eine Region nicht angeben können, indem Sie den Parameter `-Region` zu einem Befehl hinzufügen, geben Sie die Region stattdessen in einem benutzerdefinierten Endpunkt an, wie im folgenden Abschnitt gezeigt.

Angeben eines benutzerdefinierten oder nicht standardmäßigen Endpunkts

Geben Sie einen benutzerdefinierten Endpunkt als URL an, indem Sie den `-EndpointUrl` allgemeinen Parameter im folgenden Beispielformat zu Ihrem PowerShell Befehl Tools for Windows hinzufügen.

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

Im Folgenden finden Sie ein Beispiel für die Verwendung des `Get-EC2Instance`-Cmdlets. Der benutzerdefinierte Endpunkt ist in diesem Beispiel in `us-west-2` oder USA West (Oregon), aber Sie können auch jede andere unterstützte AWS -Region verwenden, einschließlich solcher Regionen, die nicht von `Get-AWSRegion` aufgezählt werden.

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

Zusätzliche Informationen

Weitere Informationen zu - AWS Regionen finden Sie unter AWS Region im AWS Referenzhandbuch zu -SDKs und Tools.

Konfigurieren einer Verbundidentität mit AWS Tools for PowerShell

Um Benutzern in der Organisation den Zugriff auf AWS-Ressourcen zu gewähren, müssen Sie eine Standardmethode für die Authentifizierung konfigurieren, die wiederholbar ist, um für Sicherheit, Auditierbarkeit, Compliance sowie die Unterstützung der Trennung von Rolle und Konto zu sorgen. Obwohl Benutzern häufig der Zugriff auf AWS-APIs gestattet wird, müssten Sie ohne API-Verbundzugriff auch AWS Identity and Access Management-(IAM)-Benutzer erstellen, was aber dem Zweck des Verbunds entgegensteht. In diesem Thema wird beschrieben, wie die SAML-Support (Security Assertion Markup Language) in den AWS Tools for PowerShell eine Verbundzugriffslösung ermöglicht.

Durch den SAML-Support in den AWS Tools for PowerShell können Sie Ihren Benutzern Verbundzugriff auf AWS-Services bereitstellen. SAML ist ein offenes Format auf XML-Basis für die Übertragung von Daten zur Benutzerauthentifizierung und -autorisierung zwischen Services, insbesondere zwischen einem Identitätsanbieter (z. B. Active-Directory-Verbundservices) und einem Service-Anbieter (wie AWS). Weitere Informationen zu SAML und ihrer Funktionsweise finden Sie unter SAML auf Wikipedia und unter SAML Technical Specifications auf der Website der Organization for the Advancement of Structured Information Standards (OASIS). Der SAML-Support in AWS Tools for PowerShell ist mit SAML 2.0 kompatibel.

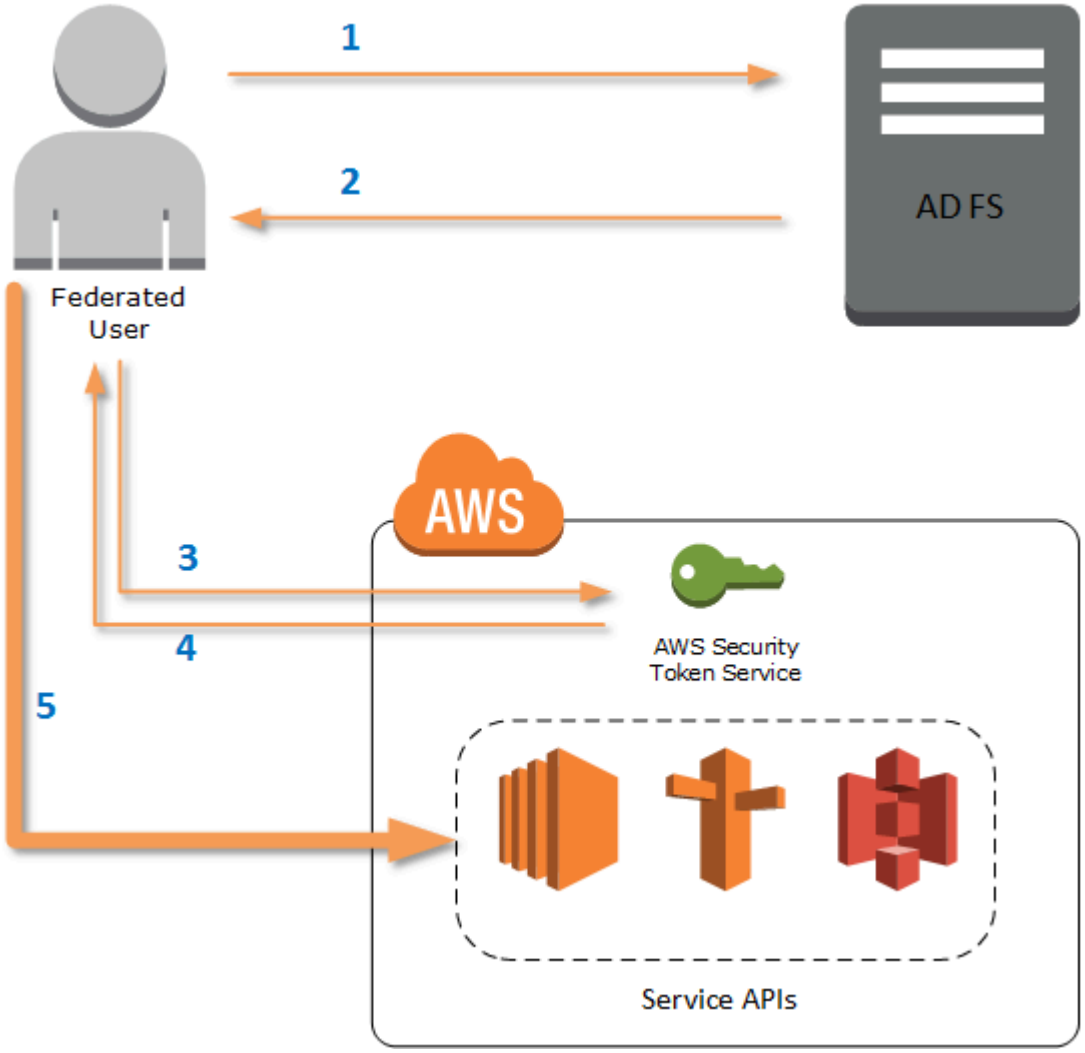
Voraussetzungen

Folgendes ist erforderlich, bevor Sie die SAML-Unterstützung erstmals nutzen können.

- Eine Verbundidentitätslösung, die richtig in das AWS-Konto integriert ist und den Konsolenzugriff nur mit den Anmeldeinformationen der Organisation ermöglicht. Wie dies mit Active Directory-Verbunddiensten erreicht werden kann, wird Informationen zum SAML 2.0-basierten Verbund im IAM-Benutzerhandbuch und im Blogbeitrag Aktivieren des Verbunds mit AWS mithilfe von Windows Active Directory, AD FS und SAML 2.0 erläutert. Der Blogbeitrag befasst sich zwar mit AD FS 2.0, die Schritte für AD FS 3.0 sind aber identisch.
- Version 3.1.31.0 oder höher der AWS Tools for PowerShell muss auf der lokalen Workstation installiert sein.

Wie ein Identitätsverbundbenutzer Verbundzugriff auf AWS-Service-APIs erhält

Der folgende Prozess beschreibt auf hoher Ebene, wie ein Active-Directory-(AD)-Benutzer mittels AD FS in einen Verbund eingefügt wird, um Zugriff auf AWS-Ressourcen zu erlangen.

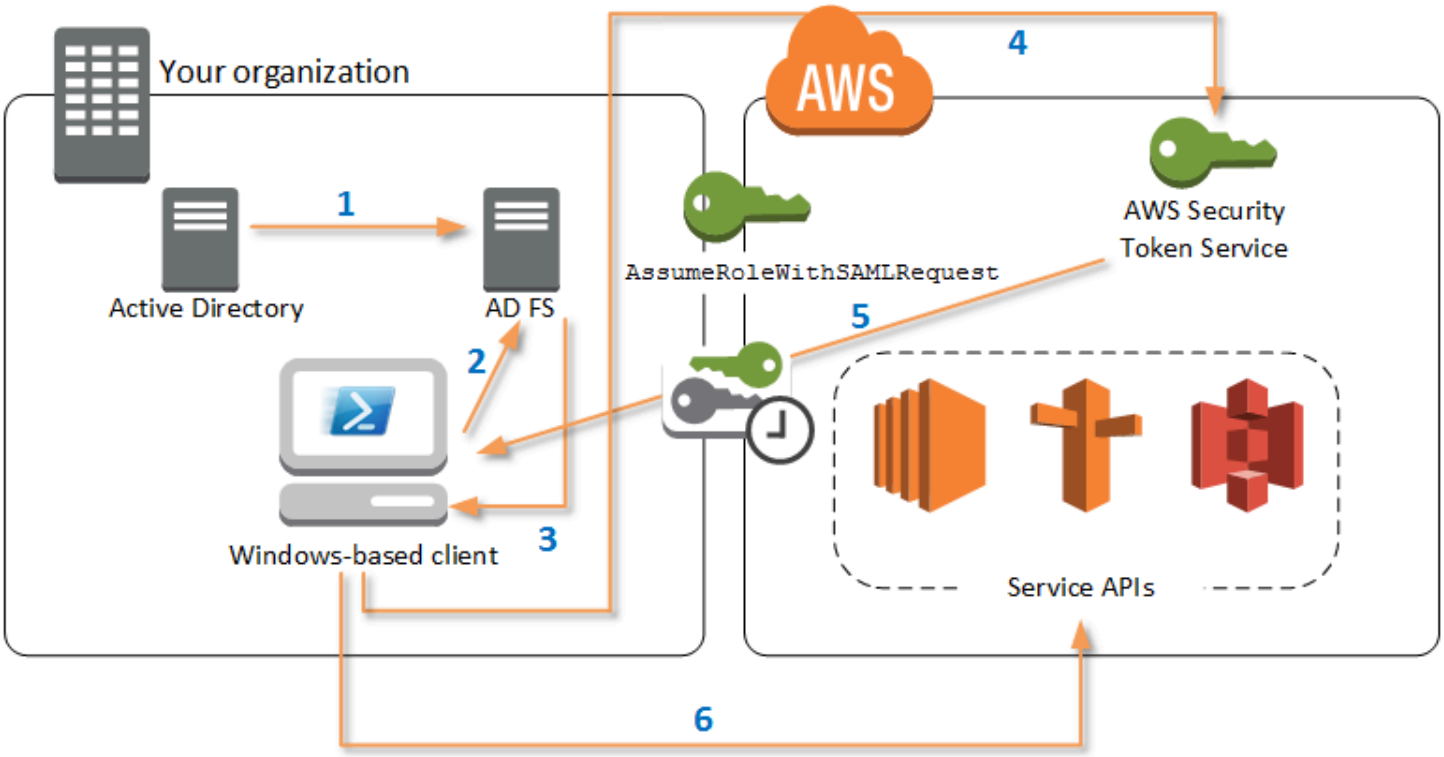


1. Der Client auf dem Computer des verbundenen Benutzers authentifiziert sich gegen AD FS.
2. Wenn die Authentifizierung erfolgreich ist, sendet AD FS eine SAML-Assertion an den Benutzer.
3. Der Client des Benutzers sendet die SAML-Assertion als Teil einer SAML-Verbundanfrage an den AWS Security Token Service (STS).
4. STS gibt eine SAML-Antwort zurück, die temporäre AWS-Anmeldeinformationen für eine Rolle enthält, die der Benutzer übernehmen kann.

5. Der Benutzer greift auf AWS-Service-APIs zu, indem er diese temporären Anmeldeinformationen in die Anfrage von AWS Tools for PowerShell einschließt.

So unterstützt SAML Arbeiten in den AWS Tools for PowerShell

In diesem Abschnitt wird beschrieben, wie AWS Tools for PowerShell-Cmdlets die Konfiguration eines Identitätsverbunds auf SAML-Basis für Benutzer ermöglicht.



1. Die AWS Tools for PowerShell nehmen die Authentifizierung für AD FS unter Verwendung der aktuellen Windows-Anmeldeinformationen oder interaktiv vor, wenn der Benutzer ein Cmdlet ausführt, das Anmeldeinformationen für Aufrufe in AWS benötigt.
2. AD FS authentifiziert den Benutzer.
3. AD FS generiert eine SAML 2.0-Authentifizierungsantwort, die eine Assertion enthält. Der Zweck der Assertion ist das Identifizieren des Benutzers und das Bereitstellen von Benutzerinformationen. AWS Tools for PowerShell extrahiert die Liste der autorisierten Rollen des Benutzers aus der SAML-Assertion.
4. Die AWS Tools for PowerShell leiten die SAML-Anfrage einschließlich des Amazon-Ressourcennamens (ARN) der angeforderten Rolle an STS weiter, indem sie die API `AssumeRoleWithSAMLRequest` aufrufen.

5. Wenn die SAML-Anforderung gültig ist, gibt STS eine Antwort mit den Werten AWS, `AccessKeyId`, `SecretAccessKey` und `SessionToken` zurück. Diese Anmeldeinformationen gelten für 3.600 Sekunden (1 Stunde).
6. Der Benutzer verfügt jetzt über gültige Anmeldeinformationen für die Arbeit mit allen AWS-Service-APIs, auf die die Rolle des Benutzers zugreifen kann. Die AWS Tools for PowerShell wenden diese Anmeldeinformationen automatisch für nachfolgende AWS-API-Aufrufe an und erneuern sie automatisch, wenn sie ablaufen.

> **Note**
>
> Wenn die Anmeldeinformationen ablaufen und neue Anmeldeinformationen benötigt werden, nehmen die AWS Tools for PowerShell die erneute Authentifizierung bei AD FS automatisch vor und rufen neue Anmeldeinformationen für eine weitere Stunde ab. Für Benutzer eines Kontos, das mit einer Domäne verknüpft ist, erfolgt dieser Prozess vollständig transparent. Für Konten, die nicht mit einer Domäne verknüpft sind, fordern die AWS Tools for PowerShell die Benutzer auf, ihre Anmeldeinformationen einzugeben, damit sie erneut authentifiziert werden können.

Verwenden der PowerShell-Cmdlets für die SAML-Konfiguration

Die AWS Tools for PowerShell enthalten zwei neue Cmdlets, die SAML-Support bereitstellen.

- `Set-AWSSamlEndpoint` konfiguriert den AD FS-Endpunkt, weist dem Endpunkt einen Anzeigenamen zu und beschreibt optional den Authentifizierungstyp des Endpunkts.
- `Set-AWSSamlRoleProfile` erstellt und bearbeitet ein Benutzerkontoprofil, das Sie einem AD FS-Endpunkt zuordnen können, der durch Angeben des Anzeigenamens identifiziert wird, den Sie für das Cmdlet `Set-AWSSamlEndpoint` bereitgestellt haben. Jedes Rollenprofil ist einer einzelnen Rolle zugeordnet, zu deren Ausführung ein Benutzer berechtigt ist.

  Wie AWS-Anmeldeinformationsprofile können Sie dem Rollenprofil einen Anzeigenamen zuordnen. Sie können diesen Anzeigenamen mit dem Cmdlet `Set-AWSCredential` oder als Wert des Parameters `-ProfileName` für alle Cmdlets verwenden, die AWS-Service-APIs aufrufen.

Öffnen Sie eine neue AWS Tools for PowerShell-Sitzung. Wenn Sie PowerShell 3.0 oder eine neuere Version ausführen, wird das AWS Tools for PowerShell-Modul automatisch importiert, sobald Sie eines der zugehörigen Cmdlets ausführen. Wenn Sie PowerShell 2.0 ausführen, müssen Sie das

Modul manuell importieren, indem Sie das Cmdlet „Import-Module“ ausführen, wie im folgenden Beispiel gezeigt.

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

Ausführen der Cmdlets **Set-AWSSamlEndpoint** und **Set-AWSSamlRoleProfile**

1. Konfigurieren Sie zunächst die Endpunkt-Einstellungen für das AD FS-System. Die einfachste Möglichkeit besteht darin, den Endpunkt wie in diesem Schritt gezeigt in einer Variablen zu speichern. Ersetzen Sie die Platzhalter für Konto-IDs und AD FS-Hostname durch die eigenen Konto-IDs und den eigenen AD FS-Hostnamen. Geben Sie den AD FS-Hostnamen im Parameter `Endpoint` an.

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

2. Führen Sie zum Erstellen des Endpunkts das Cmdlet `Set-AWSSamlEndpoint` aus. Geben Sie dabei den richtigen Wert für den Parameter `AuthenticationType` an. Gültige Werte sind `Basic`, `Digest`, `Kerberos`, `Negotiate` und `NTLM`. Wenn Sie diesen Parameter nicht angeben, lautet der Standardwert `Kerberos`.

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

   Das Cmdlet gibt den Anzeigenamen zurück, den Sie mit dem Parameter `-StoreAs` zugewiesen haben, sodass Sie ihn nutzen können, wenn Sie in der nächsten Zeile `Set-AWSSamlRoleProfile` ausführen.

3. Führen Sie jetzt das Cmdlet `Set-AWSSamlRoleProfile` aus, um den AD FS-Identitätsanbieter zu authentifizieren und die Rollen (in der SAML-Assertion) abzurufen, die der Benutzer ausführen darf.

   Das Cmdlet `Set-AWSSamlRoleProfile` verwendet die zurückgegebenen Rollen, um den Benutzer zum Auswählen einer Rolle aufzufordern, die dem angegebenen Profil zugeordnet werden soll, oder um zu validieren, ob die Rollendaten in den angegebenen Parametern vorhanden sind (andernfalls wird der Benutzer zur Auswahl aufgefordert). Wenn der Benutzer nur für eine Rolle autorisiert ist, ordnet das Cmdlet diese Rolle dem Profil automatisch zu, ohne den

Benutzer zur Auswahl aufzufordern. Es ist nicht erforderlich, Anmeldeinformationen anzugeben, um ein Profil für die Domänenverknüpfung einzurichten.

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

Alternativ können Sie für nicht mit einer Domäne verknüpfte Konten Active-Directory-Anmeldeinformationen angeben und dann eine AWS-Rolle wählen, auf die der Benutzer zugreifen kann (siehe folgende Zeile). Dies ist nützlich, wenn Sie unterschiedliche Active Directory-Benutzerkonten verwenden, um die Rollen in der Organisation zu differenzieren (z. B. Administrationsfunktionen).

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

4. In jedem Fall fordert das Cmdlet `Set-AWSSamlRoleProfile` Sie auf, die Rolle zu wählen, die im Profil gespeichert werden soll. Das folgende Beispiel zeigt zwei verfügbare Rollen: `ADFS-Dev` und `ADFS-Production`. Die IAM-Rollen werden vom AD-FS-Administrator mit Ihren AD-Anmeldeinformationen verknüpft.

```
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"):
```

Alternativ können Sie eine Rolle ohne Eingabeaufforderung angeben, indem Sie die Parameter `RoleARN`, `PrincipalARN` und optional `NetworkCredential` eingeben. Wenn die angegebene Rolle nicht in der von der Authentifizierung zurückgegebenen Assertion aufgeführt ist, wird der Benutzer aufgefordert, eine der verfügbaren Rollen auszuwählen.

```
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. Sie können Profile für alle Rollen mit nur einem Befehl erstellen, indem Sie den Parameter `StoreAllRoles` hinzufügen (siehe den folgenden Code). Beachten Sie, dass der Rollenname als Profilname verwendet wird.

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

Verwenden von Rollenprofilen zum Ausführen von Cmdlets, die AWS-Anmeldeinformationen benötigen

Um Cmdlets auszuführen, die AWS-Anmeldeinformationen benötigen, können Sie ein in der Datei mit gemeinsamen AWS-Anmeldeinformationen definiertes Rollenprofil verwenden. Übergeben Sie den Namen eines Rollenprofils an `Set-AWSCredential` (oder als Wert eines `ProfileName`-Parameters in AWS Tools for PowerShell), um automatisch temporäre AWS-Anmeldeinformationen für die im Profil beschriebene Rolle zu erhalten.

Obwohl Sie jeweils nur ein Rollenprofil verwenden, können Sie in einer Shell-Sitzung zwischen Profilen umschalten. Das Cmdlet `Set-AWSCredential` authentifiziert nicht und ruft keine Anmeldeinformationen ab, wenn Sie es selbstständig ausführen. Das Cmdlet zeichnet auf, dass Sie ein bestimmtes Rollenprofil verwenden möchten. Bis Sie ein Cmdlet ausführen, das AWS-Anmeldeinformationen benötigt, erfolgt keine Authentifizierung und es werden keine Anmeldeinformationen angefordert.

Sie können nun die temporären AWS-Anmeldeinformationen verwenden, die Sie mit dem Profil `SAMLDemoProfile` erhalten haben, um mit den AWS-Service-APIs zu arbeiten. Die folgenden Abschnitte zeigen Beispiele zur Verwendung von Rollenprofilen.

Beispiel 1: Festlegen einer Standardrolle mit **`Set-AWSCredential`**

In diesem Beispiel wird eine Standardrolle für eine AWS Tools for PowerShell-Sitzung mit `Set-AWSCredential` festgelegt. Anschließend können Sie Cmdlets ausführen, die Anmeldeinformationen benötigen und von der angegebenen Rolle autorisiert sind. In diesem Beispiel werden alle Amazon-Elastic-Compute-Cloud-Instances in der Region USA West (Oregon) aufgelistet, die dem im `Set-AWSCredential`-Cmdlet angegeben Profil zugeordnet sind.

```
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}
```

Beispiel 2: Ändern von Rollenprofilen in einer PowerShell-Sitzung

In diesem Beispiel werden alle verfügbaren Amazon-S3-Buckets im AWS-Konto der mit dem Profil `SAMLDemoProfile` verknüpften Rolle aufgelistet. Das Beispiel zeigt, dass Sie in der AWS Tools for PowerShell-Sitzung das Profil wechseln können, obwohl zuvor ein anderes Profil verwendet wurde, indem Sie für den Parameter `-ProfileName` von Cmdlets, die dies unterstützen, einen anderen Wert angeben. Dies ist eine typische Aufgabe für Administratoren, die Amazon S3 über die PowerShell-Befehlszeile verwalten.

```
PS > Get-S3Bucket -ProfileName SAMLDemoProfile

CreationDate                                                     BucketName
------------                                                     ----------
7/25/2013 3:16:56 AM                                             mybucket1
4/15/2015 12:46:50 AM                                            mybucket2
4/15/2015 6:15:53 AM                                             mybucket3
1/12/2015 11:20:16 PM                                            mybucket4
```

Beachten Sie, dass das Cmdlet `Get-S3Bucket` den Namen des Profils angibt, das durch Ausführen des Cmdlets `Set-AWSSamlRoleProfile` erstellt wurde. Dieser Befehl kann nützlich sein, wenn Sie früher in der Sitzung ein Rollenprofil festgelegt haben (z. B. durch Ausführen des Cmdlets `Set-AWSCredential`) und ein anderes Rollenprofil für das Cmdlet `Get-S3Bucket` verwenden wollen. Der Profilmanager macht dem Cmdlet `Get-S3Bucket` temporäre Anmeldeinformationen verfügbar.

Obwohl die Anmeldeinformationen nach einer Stunde ablaufen (ein von STS durchgesetztes Limit), aktualisieren die AWS Tools for PowerShell die Anmeldeinformationen automatisch, indem eine neue SAML-Assertion angefordert wird, wenn die Tools feststellen, dass die aktuellen Anmeldeinformationen abgelaufen sind.

Für Benutzer mit Domänenverknüpfung erfolgt dieser Prozess ohne Unterbrechung, weil die Windows-Identität des aktuellen Benutzers für die Authentifizierung verwendet wird. Für Benutzerkonten ohne Domänenverknüpfung zeigen die AWS Tools for PowerShell eine PowerShell-

Eingabeaufforderung an, die das Benutzerpasswort abfragt. Der Benutzer gibt Anmeldeinformationen ein, die für die erneute Authentifizierung des Benutzers und zum Abrufen einer neuen Assertion verwendet werden.

Beispiel 3: Ermitteln der Instances in einer Region

Das folgende Beispiel listet alle Amazon-EC2-Instances in der Region Asien-Pazifik (Sydney) auf, die mit dem vom Profil `ADFS-Production` verwendeten Konto verknüpft sind. Mit diesem Befehl können Sie alle Amazon-EC2-Instances in einer Region zurückgeben.

```
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
```

Weiterführende Lektüre

Allgemeine Informationen zum Implementieren des API-Verbundzugriffs finden Sie in How to Implement a General Solution for Federated API/CLI Access Using SAML 2.0.

Besuchen Sie bei Fragen oder Kommentaren zum Support die AWS-Entwickler-Foren für PowerShell-Skriptsprache oder .NET-Entwicklung.

Cmdlet-Erkennung und -Aliasse

In diesem Abschnitt wird erläutert, wie Sie von AWS Tools for PowerShell unterstützte Services auflisten, wie Sie die von AWS Tools for PowerShell zur Unterstützung dieser Services bereitgestellten Cmdlets anzeigen und wie Sie alternative Cmdlet-Namen (Aliasnamen) für den Zugriff auf diese Services finden können.

Cmdlet-Erkennung

Alle AWS-Service-Operationen (oder APIs) sind im API-Referenzhandbuch für jeden Service dokumentiert. Siehe zum Beispiel die IAM-API-Referenz. In den meisten Fällen besteht eine exakte Übereinstimmung zwischen einer AWS-Service-API und einem AWS-PowerShell-Cmdlet. Um den Cmdlet-Namen zu ermitteln, der einem AWS-Service-API-Namen entspricht, führen Sie das AWS-Cmdlet `Get-AWSCmdletName` zusammen mit dem Parameter `-ApiOperation` und dem AWS-Service-API-Namen aus. Um beispielsweise alle möglichen Cmdlet-Namen abzurufen, die auf einer verfügbaren `DescribeInstances`-AWS-Service-API basieren, führen Sie den folgenden Befehl aus:

```
PS > Get-AWSCmdletName -ApiOperation DescribeInstances

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

Der Parameter `-ApiOperation` ist der Standardparameter, sodass Sie den Parameternamen auslassen können. Das folgende Beispiel entspricht dem vorherigen:

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

Wenn Sie die Namen der API und des Services kennen, können Sie den Parameter `-Service` zusammen mit dem Cmdlet-Namenspräfix oder einem Teil des AWS-Service-Namens hinzufügen. Das Cmdlet-Namenspräfix für Amazon EC2 lautet beispielsweise `EC2`. Um den Cmdlet-Namen abzurufen, der der `DescribeInstances`-API im Amazon-EC2-Service entspricht, führen Sie einen der folgenden Befehle aus. Sie sind alle Ergebnis in der gleichen Ausgabe:

```
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
```

Bei den Parameterwerten für diese Befehle wird die Groß- und Kleinschreibung berücksichtigt.

Wenn Sie den Namen der gewünschten AWS-Service-API oder des AWS-Services nicht kennen, können Sie den Parameter `-ApiOperation` zusammen mit dem Muster für die Zuordnung und dem Parameter `-MatchWithRegex` verwenden. Um beispielsweise alle verfügbaren Cmdlet-Namen zu ermitteln, die `SecurityGroup` enthalten, führen Sie den folgenden Befehl aus:

```
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
```

Wenn Sie den Namen des AWS-Services, aber nicht den der AWS-Service-API kennen, fügen Sie den Parameter `-MatchWithRegex` sowie den Parameter `-Service` hinzu, um die Suche auf einen einzelnen Service zu beschränken. Um beispielsweise alle verfügbaren Cmdlet-Namen zu ermitteln, die `SecurityGroup` nur im Amazon-EC2-Service enthalten, führen Sie den folgenden Befehl aus

```
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
```

Wenn Sie den Namen des AWS Command Line Interface(AWS CLI)-Befehls kennen, können Sie den Parameter `-AwsCliCommand` und den gewünschten AWS CLI-Befehlsaufruf verwenden, um den Namen des Cmdlet zu erhalten, das auf der gleichen API basiert. Um beispielsweise den Cmdlet-Namen abzurufen, der dem `authorize-security-group-ingress`-Befehlsaufruf AWS CLI im Amazon-EC2-Service entspricht, führen Sie den folgenden Befehl aus:

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

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

Das Cmdlet `Get-AWSCmdletName` benötigt nur so viel vom AWS CLI-Befehlsnamen, um den Service und die AWS-API identifizieren zu können.

Um eine Liste aller Cmdlets im Tools for PowerShell Core abzurufen, führen Sie das PowerShell-Cmdlet `Get-Command` aus, wie im folgenden Beispiel dargestellt.

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

Sie können denselben Befehl mit `-Module AWSPowerShell` ausführen, um die Cmdlets in den AWS Tools for Windows PowerShell anzuzeigen.

Das Cmdlet `Get-Command` generiert die Liste der Cmdlets in alphabetischer Reihenfolge. Beachten Sie, dass die Liste standardmäßig nach PowerShell-Verb und nicht nach PowerShell-Substantiv sortiert wird.

Um die Ergebnisse stattdessen nach Service zu sortieren, führen Sie den folgenden Befehl aus:

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

Um die Cmdlets zu filtern, die vom Cmdlet `Get-Command` zurückgegeben werden, leiten Sie die Ausgabe in das PowerShell-Cmdlet `Select-String` um. Mithilfe des folgenden Befehls zeigen Sie beispielsweise die Cmdlets an, die mit AWS-Regionen arbeiten:

```
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
```

Sie können Cmdlets für einen bestimmten Service auch ermitteln, indem Sie nach dem Service-Präfix des Cmdlet-Substantivs filtern. Führen Sie `Get-AWSPowerShellVersion -ListServiceVersionInfo` aus, um die Liste der verfügbaren Servicepräfixe anzuzeigen. Das folgende Beispiel gibt Cmdlets zurück, die den Amazon-CloudWatch-Events-Service unterstützen.

```
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
```

Cmdlet-Namen und -Aliasse

Die von AWS Tools for PowerShell für einen bestimmten Service bereitgestellten Cmdlets basieren auf den Methoden, die vom AWS SDK für den betreffenden Service bereitgestellt werden. Aufgrund der obligatorischen PowerShell-Namenskonventionen kann der Name eines Cmdlets vom Namen des API-Aufrufs oder der Methode abweichen, auf denen es beruht. Beispiel: Das Cmdlet `Get-EC2Instance` basiert auf der Methode Amazon EC2`DescribeInstances`.

In manchen Fällen kann der Cmdlet-Name dem Namen einer Methode ähneln, obwohl es eine andere Funktion ausführt. Die Amazon-S3-Methode `GetObject` ruft beispielsweise ein Amazon-S3-Objekt ab. Das Cmdlet `Get-S3Object` gibt jedoch Informationen zu einem Amazon-S3-Objekt zurück, nicht das Objekt selbst.

```
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
```

Verwenden Sie zum Abrufen des S3-Objekts mit AWS Tools for PowerShell das Cmdlet `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**
>
> Die Cmdlet-Hilfe zu einem AWS-Cmdlet stellt den Namen der AWS-SDK-API bereit, auf die das Cmdlet basiert.
> Weitere Informationen zu PowerShell-Standardverben und ihren Bedeutungen finden Sie unter Genehmigte Verben für die PowerShell-Befehle.

Alle AWS-Cmdlets, die das Verb `Remove` verwenden, sowie das Cmdlet `Stop-EC2Instance`, sofern Sie den Parameter `-Terminate` hinzufügen, fordern eine Bestätigung an, bevor die Ausführung fortgesetzt wird. Mit dem Parameter `-Force` können Sie die Bestätigung umgehen.

> **Important**
>
> AWS-Cmdlets unterstützen den Schalter `-WhatIf` nicht.

Aliasnamen

Das Installationsprogramm für AWS Tools for PowerShell installiert eine Aliasdatei, die Aliasnamen für viele der AWS-Cmdlets enthält. Diese Aliasnamen sind möglicherweise intuitiver als die Cmdlet-Namen. Beispielsweise ersetzen Servicenamen und AWS-SDK-Methodennamen PowerShell-Verbs und -Substantive in manchen Aliasnamen. Ein Beispiel ist der Alias `EC2-DescribeInstances`.

Andere Aliasnamen verwenden Verben, die möglicherweise nicht den PowerShell-Konventionen entsprechen, aber trotzdem aussagekräftiger als die eigentliche Operation sein können. Die Aliasdatei ordnet beispielsweise den Alias `Get-S3Content` dem Cmdlet `Read-S3Object` zu.

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

Die Aliasdatei befindet sich im AWS Tools for PowerShell-Installationsverzeichnis. Um die Aliasnamen in die Umgebung zu laden, geben Sie die Datei in der Punkt-Quelle-Schreibweise an. Im Folgenden finden Sie ein Windows-basiertes Beispiel.

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

Für eine Linux- oder macOS-Shell könnte es folgendermaßen aussehen:

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

Führen Sie den folgenden Befehl aus, um alle AWS Tools for PowerShell-Aliasnamen anzuzeigen. Dieser Befehl verwendet den Alias ? für das PowerShell-Cmdlet `Where-Object` und die Eigenschaft `Source`, um nur nach Aliasnamen zu filtern, die aus dem Modul `AWSPowerShell.NetCore` stammen.

```
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

...
```

Um dieser Datei Ihre eigenen Aliasse hinzuzufügen, müssen Sie möglicherweise den Wert der `$MaximumAliasCount` Einstellungsvariablen von PowerShell auf einen Wert größer als 5500 setzen. Der Standardwert ist 4096. Sie können ihn auf maximal 32.768 erhöhen. Führen Sie dazu den folgenden Befehl aus.

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

Um zu überprüfen, ob Ihre Änderung erfolgreich war, geben Sie den Variablennamen ein, um den aktuellen Wert anzuzeigen.

```
PS > $MaximumAliasCount
32768
```

Pipeline-Ausführung und $AWSHistory

Für AWS-Service-Aufrufe, die Sammlungen zurückgeben, werden die Objekte in der Sammlung jetzt immer in die Pipeline enumeriert. Ergebnisobjekten, die neben der Sammlung weitere Felder enthalten und keine Steuerfelder auslagern, wurden diese Felder als Note-Eigenschaften für die Aufrufe hinzugefügt. Diese Note-Eigenschaften werden in der neuen Sitzungsvariable `$AWSHistory` protokolliert, damit Sie bei Bedarf auf diese Daten zugreifen können. Die Variable `$AWSHistory` wird im nächsten Abschnitt beschrieben.

> **Note**
>
> In Versionen von Tools for Windows PowerShell vor 1.1 wurde das Sammlungsobjekt selbst ausgegeben. Das machte die Verwendung von „foreach {$_.getenumerator()}“ zum Fortsetzen der Pipeline-Ausführung erforderlich.

Beispiele

Im folgenden Beispiel wird eine Liste der AWS-Regionen und der Amazon EC2 Machine Images (AMIs) in jeder Region zurückgegeben.

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

Im folgenden Beispiel werden alle Amazon-EC2-Instances in der aktuellen Standardregion angehalten.

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

Da Sammlungen in die Pipeline enumeriert werden, kann die Ausgabe eines gegebenen Cmdlet `$null`, ein einzelnes Objekt oder eine Sammlung sein. Falls es sich um eine Sammlung handelt, können Sie die Eigenschaft `.Count` verwenden, um die Größe der Sammlung zu bestimmen. Die Eigenschaft `.Count` ist jedoch nicht vorhanden, wenn nur ein Objekt ausgegeben wird. Wenn das Skript auf konsistente Weise bestimmen muss, wie viele Objekte ausgegeben wurden, können Sie die Eigenschaft `EmittedObjectsCount` des letzten Befehlswerts in `$AWSHistory` überprüfen.

$AWSHistory

Um die Pipeline-Ausführung besser zu unterstützen, wird die Ausgabe von AWS-Cmdlets nicht so umgeformt, dass sie die Antwort- und Ergebnis-Instances des Services als Note-Eigenschaften im ausgegebenen Sammlungsobjekt enthält. Stattdessen wird die Sammlung bei Aufrufen, die eine einzelne Sammlung ausgeben, in die PowerShell-Pipeline enummeriert. Das bedeutet, dass die AWS-SDK-Antwort- und Ergebnisdaten nicht in der Pipe vorliegen können, weil es kein Sammlungsobjekt gibt, an das sie angehängt werden können.

Obwohl die meisten Benutzer diese Daten wahrscheinlich nicht benötigen werden, sind sie für Diagnosezwecke nützlich, weil Sie ermitteln können, was genau an die vom Cmdlet veranlassten AWS-Service-Aufrufe übergeben und was von diesen empfangen wurde.

Ab Version 1.1 sind diese und weitere Daten in einer neuen Shell-Variable namens `$AWSHistory` verfügbar. Diese Variable protokolliert AWS-Cmdlet-Aufrufe sowie die Service-Antworten, die als Resultat der Aufrufe empfangen wurden. Optional kann dieses Protokoll so konfiguriert werden, dass auch die Service-Anforderungen aufgezeichnet werden, die von den Cmdlets ausgegeben werden. Zusätzliche nützliche Daten wie die Gesamtausführungszeit des Cmdlets können ebenfalls aus jedem Eintrag abgerufen werden. Aus Sicherheitsgründen werden Anfragen und Antworten, die vertrauliche Daten enthalten, standardmäßig nicht aufgezeichnet. Der Verlauf kann jedoch so konfiguriert werden,

dass er dieses Verhalten bei Bedarf außer Kraft setzt. Weitere Informationen finden Sie im folgenden `Set-AWSHistoryConfiguration`-Cmdlet.

Jeder Eintrag in der Liste `$AWSHistory.Commands` hat den Typ `AWSCmdletHistory`. Dieser Typ besteht aus den folgenden nützlichen Elementen:

CmdletName

Name des Cmdlets.

CmdletStart

DateTime-Wert der Cmdlet-Ausführung.

CmdletEnd

DateTime-Wert des Cmdlet-Ausführungsendes.

Anforderungen

Wenn die Protokollierung von Anforderungen aktiviert ist, ist dies eine Liste der letzten Service-Anforderungen.

Antworten

Liste der zuletzt empfangenen Service-Antworten.

LastServiceResponse

Gibt die letzte Service-Antwort zurück.

LastServiceRequest

Gibt die letzte Service-Anforderung zurück (sofern verfügbar).

Beachten Sie, dass die Variable `$AWSHistory` erst erstellt wird, wenn ein AWS-Cmdlet verwendet wird, das einen Service aufruft. Bis zu diesem Zeitpunkt wird das Element als $null ausgewertet.

> **Note**
>
> Frühere Versionen von Tools for Windows PowerShell haben Daten zu Service-Antworten als `Note`-Eigenschaften im zurückgegebenen Objekt ausgegeben. Diese Daten befinden sich jetzt in den Antworteinträgen, die für jeden Aufruf in der Liste protokolliert werden.

Set-AWSHistoryConfiguration

Ein Cmdlet-Aufruf kann null oder mehr Einträge für Service-Anforderungen und -Antworten enthalten. Um die Auswirkungen auf den Arbeitsspeicher zu begrenzen, enthält `$AWSHistory` standardmäßig eine Liste nur der letzten fünf Cmdlet-Ausführungen und für diese jeweils die letzten fünf Service-Antworten (und, sofern aktiviert, die letzten fünf Service-Anforderungen). Sie können diese Standardbegrenzungen ändern, indem Sie das Cmdlet `Set-AWSHistoryConfiguration` ausführen. Mit diesem Cmdlet können Sie die Größe der Liste steuern und festlegen, ob auch Service-Anforderungen protokolliert werden:

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

Alle Parameter sind optional.

Der Parameter `MaxCmdletHistory` legt die maximale Anzahl von Cmdlets fest, die zu einem bestimmten Zeitpunkt verfolgt werden. Ein Wert von 0 deaktiviert das Protokollieren von AWS-Cmdlet-Aktivitäten. Der Parameter `MaxServiceCallHistory` legt die maximale Anzahl von Service-Antworten (und/oder -Anforderungen) fest, die pro Cmdlet verfolgt werden. Der Parameter `RecordServiceRequests` aktiviert (sofern angegeben) die Verfolgung der Service-Anforderungen für die Cmdlets. Der `IncludeSensitiveData`-Parameter aktiviert, sofern angegeben, die Nachverfolgung von Service-Reaktionen und -Anfragen (sofern diese nachverfolgt werden), die vertrauliche Daten für jedes Cmdlet enthalten.

Wenn `Set-AWSHistoryConfiguration` ohne Parameter ausgeführt wird, wird die aktivierte Anforderungsprotokollierung einfach deaktiviert. Die derzeitigen Listengrößen bleiben unverändert.

Führen Sie zum Löschen aller Einträge aus der aktuellen Protokollliste das Cmdlet `Clear-AWSHistory` aus.

Beispiele für **`$AWSHistory`**

Enummerieren Sie die Details der in der Liste vorhandenen AWS-Cmdlets in die Pipeline.

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

Zugriff auf die Details des zuletzt ausgeführten AWS-Cmdlets:

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

Greifen Sie auf die Details der letzten Service-Antwort zu, die vom zuletzt ausgeführten AWS-Cmdlet empfangen wurde. Wenn ein AWS-Cmdlet die Ausgabe auslagert, veranlasst es ggf. mehrere Service-Aufrufe, um alle Daten oder die maximal zulässige Datenmenge (festgelegt mit den Parametern des Cmdlets) abzurufen.

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

Greifen Sie auf die Details der letzten Anforderung zu (wiederum gilt, dass ein Cmdlet mehr als eine Anforderung ausgeben kann, wenn die Daten für den Benutzer ausgelagert werden). Liefert $null, sofern die Verfolgung von Service-Anforderungen nicht aktiviert ist.

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

Automatische seitenweise Verarbeitung bis zur Fertigstellung für Operationen, die mehrere Seiten zurückgeben

Für Service-APIs, die standardmäßig eine maximale Anzahl zurückgegebener Objekte für einen gegebenen Aufruf festlegen oder auslagerbare Ergebnismengen unterstützen, nehmen alle Cmdlets standardmäßig eine seitenweise Verarbeitung bis zur Fertigstellung vor. Jedes Cmdlet gibt so viele Aufrufe wie nötig aus, um die gesamte Datenmenge an die Pipeline zurückzugeben.

Im folgenden Beispiel, das `Get-S3Object` verwendet, enthält die Variable `$c` `S3Object`-Instances für jeden Schlüssel im Bucket `test`. Dabei kann es sich um eine sehr große Datenmenge handeln.

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

Wenn Sie die Kontrolle über die Menge der zurückgegebenen Daten behalten möchten, können Sie Parameter für individuelle Cmdlets (z. B. `MaxKey` für `Get-S3Object`) verwenden oder die Auslagerung explizit selbst handhaben – durch Verwendung einer Kombination aus Auslagerungsparametern von Cmdlets und der Daten in der Variablen `$AWSHistory`, um die nächsten Tokendaten des Services abzurufen. Im Folgenden werden die "MaxKeys"-Parameter verwendet, um die Anzahl der `S3Object`-Instances auf maximal die ersten 500 im Bucket gefundenen Instances zu begrenzen.

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

Um in Erfahrung zu bringen, ob mehr Daten verfügbar waren, aber nicht zurückgegeben wurden, verwenden Sie den Sitzungsvariableneintrag `$AWSHistory`, in dem die vom Cmdlet getätigten Service-Aufrufe protokolliert wurden.

Wenn der folgende Ausdruck als $true ausgewertet wird, können Sie die Markierung `next` für die nächste Ergebnismenge mit `$AWSHistory.LastServiceResponse.NextMarker` ermitteln:

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

Um die Auslagerung manuell mit `Get-S3Object` zu steuern, verwenden Sie eine Kombination der Parameter `MaxKey` und `Marker` für das Cmdlet und die Note-Eigenschaften `IsTruncated`/`NextMarker` der letzten protokollierten Antwort. Im folgenden Beispiel enthält die Variable `$c` maximal 500 `S3Object`-Instances für die nächsten 500 Objekte, die hinter der angegebenen Schlüsselpräfixmarkierung im Bucket gefunden werden.

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

Auflösung von Anmeldeinformationen und Profilen

Suchreihenfolge für Anmeldeinformationen

Wenn Sie einen Befehl ausführen, suchen die AWS Tools for PowerShell in der folgenden Reihenfolge nach Anmeldeinformationen. Dieser Vorgang wird beendet, wenn verwendbare Anmeldeinformationen gefunden wurden.

1. Literale Anmeldeinformationen, die als Parameter in der Befehlszeile eingebettet sind.

   Wir empfehlen dringend, Profile zu verwenden, anstatt literale Anmeldeinformationen in die Befehlszeile einzugeben.
2. Ein festgelegter Profilname oder Profilspeicherort.
   - Wenn Sie nur einen Profilnamen angeben, wird das angegebene Profil bei Ausführung des Befehls im AWS-SDK-Speicher gesucht. Ist es dort nicht vorhanden, wird das angegebene Profil in der Datei mit gemeinsamen AWS-Anmeldeinformationen im Standardspeicherort gesucht.
   - Wenn Sie nur einen Profilspeicherort angeben, wird bei Ausführung des Befehls das `default`-Profil in dieser Datei mit Anmeldeinformationen gesucht.

- Wenn Sie sowohl einen Namen als auch einen Speicherort angeben, wird bei Ausführung des Befehls nach dem angegebenen Profil in dieser Datei mit Anmeldeinformationen gesucht.

Wenn der angegebene Profilname oder Speicherort nicht gefunden wird, löst der Befehl eine Ausnahme aus. Die folgenden Suchschritte werden nur durchgeführt, wenn Sie kein Profil und keinen Speicherort angegeben haben.

3. Mit dem Parameter `-Credential` angegebene Anmeldeinformationen.
4. Das Sitzungsprofil, sofern vorhanden.
5. Das Standardprofil in der folgenden Reihenfolge:
   a. Das Profil `default` im AWS SDK-Speicher.
   b. Das Profil `default` in der Datei mit gemeinsamen AWS-Anmeldeinformationen.
   c. Das Profil `AWS PS Default` im AWS SDK-Speicher.
6. Wenn der Befehl auf einer Amazon-EC2-Instance ausgeführt wird, die für die Verwendung einer IAM-Rolle konfiguriert ist, werden die temporären Anmeldeinformationen der EC2-Instance über das Instance-Profil aufgerufen.

   Weitere Informationen zur Verwendung von IAM-Rollen für Amazon-EC2-Instances finden Sie unter AWS SDK for .NET.

Wenn die angegebenen Anmeldeinformationen im Rahmen dieser Suche nicht gefunden werden, löst der Befehl eine Ausnahme aus.

Zusätzliche Informationen über Benutzer und Rollen

Um auf AWS Tools für PowerShell-Befehle ausführen zu können, benötigen Sie eine Kombination aus Benutzern, Berechtigungssätzen und Servicerollen, die für Ihre Aufgaben geeignet ist.

Welche spezifischen Benutzer, Berechtigungssätze und Servicerollen Sie erstellen und wie Sie sie verwenden, hängt von Ihren Anforderungen ab. Im Folgenden finden Sie einige zusätzliche Informationen darüber, warum sie verwendet werden können und wie sie erstellt werden.

Benutzer und Berechtigungssätze

Es ist zwar möglich, ein IAM-Benutzerkonto mit langfristigen Anmeldeinformationen für den Zugriff auf AWS-Services zu verwenden, dies ist jedoch keine bewährte Methode mehr und sollte vermieden

werden. Selbst bei der Entwicklung hat es sich bewährt, Benutzer und Berechtigungssätze in AWS IAM Identity Center zu erstellen und temporäre Anmeldeinformationen zu verwenden, die von einer Identitätsquelle bereitgestellt werden.

Für die Entwicklung können Sie den Benutzer verwenden, den Sie erstellt haben oder den Sie in Tool-Authentifizierung konfigurieren erhalten haben. Wenn Sie über die entsprechenden AWS Management Console-Berechtigungen verfügen, können Sie auch verschiedene Berechtigungssätze mit der geringsten Berechtigung für diesen Benutzer erstellen oder neue Benutzer speziell für Entwicklungsprojekte erstellen, indem Sie Berechtigungssätze mit der geringsten Berechtigung bereitstellen. Die Vorgehensweise, die Sie auswählen (sofern Sie dies tun), hängt von Ihren Umständen ab.

Weitere Informationen zu diesen Benutzern und Berechtigungssätzen sowie zu deren Erstellung finden Sie unter Authentifizierung und Zugriff im Referenzhandbuch für AWS SDKs und Tools sowie unter Erste Schritte im Benutzerhandbuch für AWS IAM Identity Center.

Servicerollen

Sie können eine AWS-Servicerolle einrichten, um im Namen von Benutzern auf AWS-Services zuzugreifen. Diese Art des Zugriffs ist geeignet, wenn mehrere Personen Ihre Anwendung remote ausführen, z. B. auf einer Amazon-EC2-Instance, die Sie für diesen Zweck erstellt haben.

Das Verfahren zur Erstellung einer Servicerolle ist je nach Situation unterschiedlich, sieht aber im Wesentlichen wie folgt aus.

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie Roles (Rollen) und anschließend Create role (Rolle erstellen).
3. Wählen Sie AWS-Service aus, suchen und wählen Sie (zum Beispiel) EC2 und dann (zum Beispiel) den EC2-Anwendungsfall aus.
4. Wählen Sie Weiter und die entsprechenden Richtlinien für die AWS-Services aus, die Ihre Anwendung verwenden soll.

> **Warning**
>
> Wählen Sie NICHT die AdministratorAccess-Richtlinie aus, da diese Richtlinie nahezu unbegrenzte Lese- und Schreibberechtigungen für Ihr Konto gewährt.

5. Wählen Sie Next (Weiter). Geben Sie einen Rollennamen, eine Beschreibung und alle gewünschten Tags ein.

   Informationen zu Tags finden Sie unter Zugriffssteuerung mit AWS-Ressourcen-Tags im IAM-Benutzerhandbuch.

6. Wählen Sie Create role (Rolle erstellen) aus.

Allgemeine Informationen zu IAM-Rollen finden Sie unter IAM-Identitäten (Benutzer, Benutzergruppen und Rollen) im IAM-Benutzerhandbuch. Ausführliche Informationen zu Rollen finden Sie im Thema IAM-Rollen.

Verwenden von Legacy-Anmeldeinformationen

Die Themen in diesem Abschnitt enthalten Informationen zur Verwendung von lang- oder kurzfristigen Anmeldeinformationen ohne Verwendung von AWS IAM Identity Center.

> **Warning**
>
> Um Sicherheitsrisiken zu vermeiden, sollten Sie IAM-Benutzer nicht zur Authentifizierung verwenden, wenn Sie eigens entwickelte Software entwickeln oder mit echten Daten arbeiten. Verwenden Sie stattdessen den Verbund mit einem Identitätsanbieter wie AWS IAM Identity Center.

> **Note**
>
> Die Informationen in diesen Themen beziehen sich auf Situationen, in denen Sie kurz- oder langfristige Anmeldeinformationen manuell abrufen und verwalten müssen. Weitere Informationen zu kurz- und langfristigen Anmeldeinformationen finden Sie unter Andere Authentifizierungsmethoden im Referenzhandbuch für AWS SDKs und Tools.
> Als bewährte Sicherheitsmethode verwenden Sie AWS IAM Identity Center, wie unter Tool-Authentifizierung konfigurieren beschrieben.

Wichtige Warnhinweise und Richtlinien für Anmeldeinformationen

Warnhinweise für Anmeldeinformationen

- Verwenden Sie NICHT die Root-Anmeldeinformationen Ihres Kontos, um auf Ihre AWS-Ressourcen zuzugreifen. Diese Anmeldeinformationen bieten uneingeschränkten Zugriff auf Konten und können nur schwer widerrufen werden.
- Fügen Sie in Ihren Befehlen oder Skripten KEINE literalen Zugriffsschlüssel oder Anmeldeinformationen ein. Andernfalls besteht die Gefahr, dass Sie Ihre Anmeldeinformationen versehentlich preisgeben.
- Beachten Sie, dass die Anmeldeinformationen in der freigegebenen AWS-Datei `credentials` im Klartext gespeichert werden.

Zusätzliche Hinweise zur sicheren Verwaltung von Anmeldeinformationen

Eine allgemeine Erläuterung der sicheren Verwaltung von AWS-Anmeldeinformationen finden Sie unter AWS-Sicherheitsanmeldedaten in der Allgemeine AWS-Referenz und unter Bewährte Sicherheitsmethoden und Anwendungsfälle im IAM-Benutzerhandbuch. Berücksichtigen Sie zusätzlich zu diesen Informationen Folgendes:

- Erstellen Sie zusätzliche Benutzer, z. B. Benutzer in IAM Identity Center, und verwenden Sie deren Anmeldeinformationen anstelle Ihrer AWS-Root-Benutzeranmeldeinformationen. Anmeldeinformationen für andere Benutzer können bei Bedarf widerrufen werden oder sind temporärer Natur. Darüber hinaus können Sie auf jeden Benutzer eine Richtlinie anwenden, die nur den Zugriff auf bestimmte Ressourcen und Aktionen vorsieht und so eine Einstellung der Rechte mit den geringsten Berechtigungen einräumt.

- Verwenden Sie IAM-Rollen für Aufgaben in Verbindung mit Aufgaben von Amazon Elastic Container Service (Amazon ECS).

- Verwenden Sie IAM-Rollen für Anwendungen, die auf Amazon-EC2-Instances ausgeführt werden.

Themen

Verwenden von AWS-Anmeldeinformationen

Jeder AWS Tools for PowerShell-Befehl muss AWS-Anmeldeinformationen enthalten, mit denen die entsprechende Webserviceanfrage verschlüsselt signiert wird. Sie können Anmeldeinformationen pro Befehl, pro Sitzung oder für alle Sitzungen angeben.

> **Warning**
>
> Um Sicherheitsrisiken zu vermeiden, sollten Sie IAM-Benutzer nicht zur Authentifizierung verwenden, wenn Sie eigens entwickelte Software entwickeln oder mit echten Daten arbeiten. Verwenden Sie stattdessen den Verbund mit einem Identitätsanbieter wie AWS IAM Identity Center.

> **Note**
>
> Die Informationen in diesem Thema beziehen sich auf Situationen, in denen Sie kurz- oder langfristige Anmeldeinformationen manuell abrufen und verwalten müssen. Weitere Informationen zu kurz- und langfristigen Anmeldeinformationen finden Sie unter Andere Authentifizierungsmethoden im Referenzhandbuch für AWS SDKs und Tools.
> Als bewährte Sicherheitsmethode verwenden Sie AWS IAM Identity Center, wie unter Tool-Authentifizierung konfigurieren beschrieben.

Es ist ratsam, die Anmeldeinformationen nicht literal in einem Befehl anzugeben, um deren Offenlegung zu vermeiden. Erstellen Sie stattdessen ein Profil für jeden Satz von Anmeldeinformationen, den Sie verwenden möchten, und speichern Sie es an einem der beiden Speicherorte für Anmeldeinformationen. Geben Sie den Namen des betreffenden Profils im Befehl an, damit die AWS Tools for PowerShell die zugehörigen Anmeldeinformationen abrufen. Allgemeine Informationen zum sicheren Verwalten von AWS-Anmeldeinformationen finden Sie unter Bewährte Methoden für die Verwaltung von AWS-Zugriffsschlüsseln im Allgemeine Amazon Web Services-Referenz.

> **Note**
>
> Sie benötigen ein AWS-Konto, um Anmeldeinformationen abzurufen und die AWS Tools for PowerShell verwenden zu können. Informationen zum Erstellen eines AWS-Kontos finden Sie

unter Erste Schritte: Verwenden Sie AWS zum ersten Mal? im Referenzhandbuch für AWS Account Management.

Themen

Speicherorte für Anmeldeinformationen

AWS Tools for PowerShell können zwei alternative Speicherorte für Anmeldeinformationen verwenden.

- Den AWS SDK-Speicher, der die Anmeldeinformationen verschlüsselt und im Basisordner speichert. In Windows befindet sich dieser Speicher unter: `C:\Users\`*`username`*`\AppData\Local\AWSToolkit\RegisteredAccounts.json`.

  Der AWS SDK for .NET und Toolkit for Visual Studio auch den AWS SDK-Speicher.
- Die Datei mit gemeinsamen Anmeldeinformationen, die sich ebenfalls im Basisordner befindet, in der die Daten allerdings als Klartext gespeichert werden.

  Die Datei mit Anmeldeinformationen wird standardmäßig in folgendem Pfad gespeichert:

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

  Die AWS-SDKs und die AWS Command Line Interface können die Datei mit Anmeldeinformationen auch verwenden. Wenn Sie ein Skript außerhalb des AWS-Benutzerkontextes ausführen möchten, müssen Sie die Datei mit den Anmeldeinformationen in einen Speicherort kopieren, auf den alle Benutzerkonten (lokales System und Benutzer) zugreifen können.

Verwalten von Profilen

Mit Profilen können Sie verschiedene Anmeldeinformationen mit AWS Tools for PowerShell referenzieren. Sie können AWS Tools for PowerShell-Cmdlets verwenden, um Ihre Profile im AWS SDK-Speicher zu verwalten. Sie können Profile im AWS-SDK-Speicher auch mit dem Toolkit for Visual Studio oder programmgesteuert über das AWS SDK for .NET verwalten. Informationen zum Verwalten von Profilen in der Datei mit Anmeldeinformationen finden Sie unter Bewährte Methoden für die Verwaltung von AWS-Zugriffsschlüsseln.

Hinzufügen eines neuen Profils

Um ein neues Profil zum AWS-SDK-Speicher hinzuzufügen, führen Sie den Befehl `Set-AWSCredential` aus. Dieser speichert Ihren Zugriffsschlüssel und Ihren geheimen Schlüssel in der Standarddatei mit Anmeldeinformationen unter dem von Ihnen angegebenen Profilnamen.

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

- `-AccessKey` – Die Zugriffsschlüssel-ID.
- `-SecretKey`: Geheimer Schlüssel
- `-StoreAs`: Profilname, der eindeutig sein muss Verwenden Sie den Namen `default`, um das Standardprofil anzugeben.

Aktualisieren eines Profils

Der AWS-SDK-Speicher muss manuell verwaltet werden. Wenn Sie später die Anmeldeinformationen für den Service ändern (z. B. über die IAM-Konsole), wird beim Ausführen eines Befehls mit den lokal gespeicherten Anmeldeinformationen die folgende Fehlermeldung angezeigt:

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

Sie können ein Profil aktualisieren, indem Sie für dieses den Befehl `Set-AWSCredential` erneut eingeben und dabei den neuen Zugriffsschlüssel und den geheimen Schlüssel übergeben.

Auflisten von Profilen

Sie können die aktuelle Namensliste mit dem folgenden Befehl überprüfen. In diesem Beispiel hat ein Benutzer namens Shirley Zugriff auf drei Profile, die alle in der Datei mit gemeinsamen Anmeldeinformationen (`~/.aws/credentials`) gespeichert sind.

```
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
```

Entfernen eines Profils

Verwenden Sie den folgenden Befehl, um ein Profil zu entfernen, das Sie nicht mehr benötigen.

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

Der Parameter `-ProfileName` gibt das Profil an, das Sie löschen möchten.

Der veraltete Befehl Clear-AWSCredential ist aus Gründen der Abwärtskompatibilität weiterhin verfügbar, `Remove-AWSCredentialProfile` wird jedoch bevorzugt.

Festlegen von Anmeldeinformationen

Anmeldeinformationen können auf mehrere Arten festgelegt werden. Die bevorzugte Methode besteht darin, ein Profil zu identifizieren, anstatt literale Anmeldeinformationen in die Befehlszeile einzubinden. Die AWS Tools for PowerShell suchen das Profil mithilfe einer Suchreihenfolge, die unter Suchreihenfolge für Anmeldeinformationen beschrieben wird.

Unter Windows werden im AWS SDK-Speicher gespeicherte AWS-Anmeldeinformationen mit der angemeldeten Windows-Benutzeridentität verschlüsselt. Sie können nicht mit einem anderen Konto entschlüsselt oder auf einem Gerät verwendet werden, das sich von dem Gerät unterscheidet, auf dem sie ursprünglich erstellt wurden. Wenn Sie Aufgaben durchführen möchten, für die die Anmeldeinformationen eines anderen Benutzers erforderlich sind, z. B. ein Benutzerkonto, unter dem eine geplante Aufgabe ausgeführt wird, erstellen Sie wie im vorherigen Abschnitt beschrieben ein Anmeldeprofil, das Sie zum Anmelden als der entsprechende Benutzer auf dem Computer verwenden können. Melden Sie sich als Benutzer mit der Aufgabe an, um die Schritte zum Einrichten

von Anmeldeinformationen abzuschließen, und erstellen Sie ein Profil, das für diesen Benutzer funktioniert. Melden Sie sich dann ab und mit Ihren eigenen Anmeldeinformationen erneut an, um die geplante Aufgabe einzurichten.

> **Note**
>
> Geben Sie das Profil mit dem Parameter `-ProfileName` an. Dieser Parameter entspricht dem Parameter `-StoredCredentials` in früheren Versionen von AWS Tools for PowerShell. Aus Gründen der Abwärtskompatibilität wird `-StoredCredentials` weiterhin unterstützt.

Standardprofil (empfohlen)

Alle AWS SDKs und Management-Tools können Ihre Anmeldeinformationen automatisch auf Ihrem lokalen Computer finden, wenn die Anmeldeinformationen in einem Profil mit dem Namen `default` gespeichert sind. Wenn Sie beispielsweise über ein Profil mit dem Namen `default` auf dem lokalen Computer verfügen, müssen Sie weder das Cmdlet `Initialize-AWSDefaultConfiguration` noch das Cmdlet `Set-AWSCredential` ausführen. Die Tools verwenden automatisch die Zugriffs- und geheimen Schlüsseldaten, die in diesem Profil gespeichert sind. Um eine andere AWS-Region als Ihre Standardregion zu verwenden (die Ergebnisse von `Get-DefaultAWSRegion`), können Sie `Set-DefaultAWSRegion` ausführen und eine Region angeben.

Wenn Ihr Profil nicht `default` heißt, aber Sie es als Standardprofil für die aktuelle Sitzung verwenden möchten, führen Sie `Set-AWSCredential` aus, um es als Standardprofil festzulegen.

Bei der Ausführung von `Initialize-AWSDefaultConfiguration` können Sie zwar ein Standardprofil für alle PowerShell-Sitzungen festlegen, aber das Cmdlet lädt Anmeldeinformationen aus Ihrem Profil mit einem benutzerdefinierten Namen, überschreibt jedoch das `default`-Profil mit dem benannten Profil.

Wir empfehlen, `Initialize-AWSDefaultConfiguration` nicht auszuführen, es sei denn, Sie führen eine PowerShell-Sitzung auf einer Amazon-EC2-Instance aus, die nicht mit einem Instance-Profil gestartet wurde, und Sie möchten das Anmeldeinformationsprofil manuell einrichten. Beachten Sie, dass das Anmeldeinformationsprofil in diesem Szenario keine Anmeldeinformationen enthalten würde. Das Anmeldeinformationsprofil, das sich aus der Ausführung von `Initialize-AWSDefaultConfiguration` auf einer EC2-Instance ergibt, speichert Anmeldeinformationen nicht direkt. Stattdessen verweist es auf Instance-Metadaten (die temporäre Anmeldeinformationen

bereitstellen, die automatisch rotiert werden). Es speichert jedoch die Region der Instance. Ein anderes Szenario, in dem es erforderlich sein könnte, `Initialize-AWSDefaultConfiguration` auszuführen, liegt vor, wenn Sie einen Aufruf für eine andere Region ausführen möchten, als für die Region, in der die Instance ausgeführt wird. Dieser Befehl überschreibt permanent die Region, die in den Instance-Metadaten gespeichert ist.

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

> **Note**
>
> Die Standard-Anmeldeinformationen werden im AWS-SDK-Speicher unter dem Profilnamen `default` gespeichert. Der Befehl überschreibt ein evtl. vorhandenes Profil mit diesem Namen.

Wenn Ihre EC2-Instance mit einem Instance-Profil gestartet wurde, ruft PowerShell die AWS-Anmeldeinformationen und Regionsinformationen automatisch aus dem Instance-Profil ab. Es ist nicht notwendig, `Initialize-AWSDefaultConfiguration` auszuführen. Die Ausführung des Cmdlets `Initialize-AWSDefaultConfiguration` auf einer mit einem Instance-Profil gestarteten EC2-Instance ist nicht erforderlich, da es dieselben Instance-Profildaten verwendet, die PowerShell bereits standardmäßig verwendet.

Sitzungsprofil

Mit dem Befehl `Set-AWSCredential` können Sie ein Standardprofil für eine bestimmte Sitzung festlegen. Dieses Profil setzt während der Sitzung jedes Standardprofil außer Kraft. Dies wird empfohlen, wenn Sie in Ihrer Sitzung anstelle des aktuellen `default`-Profils ein Profil mit benutzerdefiniertem Namen verwenden möchten.

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

> **Note**
>
> In den Versionen von Tools for Windows PowerShell vor 1.1 funktionierte das Cmdlet `Set-AWSCredential` nicht richtig und überschrieb das mit „MyProfileName“ angegebene Profil. Wir empfehlen daher die Verwendung einer neueren Version von Tools for Windows PowerShell.

Befehlsprofil

Bei einzelnen Befehlen können Sie den Parameter `-ProfileName` hinzufügen, um ein Profil anzugeben, das nur für diesen Befehl gilt. Dieses Profil überschreibt alle Standard- oder Sitzungsprofile, wie im folgenden Beispiel gezeigt.

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

> **Note**
>
> Wenn Sie ein Standard- oder Sitzungsprofil festlegen, können Sie mit dem Parameter `-Region` eine Standard- oder Sitzungsregion überschreiben. Weitere Informationen finden Sie unter Angeben von AWS Regionen. Im folgenden Beispiel werden ein Standardprofil und eine Standardregion festgelegt.
>
> ```
> PS > Initialize-AWSDefaultConfiguration -ProfileName MyProfileName -Region us-west-2
> ```

Standardmäßig wird davon ausgegangen, dass sich die Datei mit gemeinsamen AWS-Anmeldeinformationen im Basisordner des Benutzers befindet (`C:\Users\username\.aws` unter Windows, `~/.aws` unter Linux). Um eine Datei mit Anmeldeinformationen an einem anderen Speicherort anzugeben, fügen Sie den Parameter `-ProfileLocation` ein und geben Sie den Pfad der Datei mit Anmeldeinformationen an. Im folgenden Beispiel wird eine andere als die Standard-Anmeldeinformationsdatei für einen bestimmten Befehl angegeben.

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

> **Note**
>
> Wenn Sie ein PowerShell-Skript zu einem Zeitpunkt ausführen möchten, zu dem Sie normalerweise nicht bei AWS angemeldet sind (um damit beispielsweise eine geplante Aufgabe außerhalb der normalen Arbeitszeit auszuführen), fügen Sie beim Festlegen des gewünschten Profils den Parameter `-ProfileLocation` hinzu und weisen diesem den Pfad der Datei mit den Anmeldeinformationen zu. Um sicherzustellen, dass das AWS Tools for PowerShell-Skript mit den richtigen Anmeldeinformationen ausgeführt wird,

sollten Sie den Parameter immer `-ProfileLocation` hinzufügen, wenn das Skript in einem Kontext oder Prozess außerhalb eines AWS-Kontos aufgerufen wird. Sie können die Anmeldeinformationsdatei auch an einen Speicherort kopieren, der für das lokale System oder das andere Konto zugänglich ist, das von den Skripts zum Ausführen von Aufgaben verwendet wird.

Suchreihenfolge für Anmeldeinformationen

Wenn Sie einen Befehl ausführen, suchen die AWS Tools for PowerShell in der folgenden Reihenfolge nach Anmeldeinformationen. Dieser Vorgang wird beendet, wenn verwendbare Anmeldeinformationen gefunden wurden.

1. Literale Anmeldeinformationen, die als Parameter in der Befehlszeile eingebettet sind.

   Wir empfehlen dringend, Profile zu verwenden, anstatt literale Anmeldeinformationen in die Befehlszeile einzugeben.
2. Ein festgelegter Profilname oder Profilspeicherort.
   - Wenn Sie nur einen Profilnamen angeben, wird das angegebene Profil bei Ausführung des Befehls im AWS-SDK-Speicher gesucht. Ist es dort nicht vorhanden, wird das angegebene Profil in der Datei mit gemeinsamen AWS-Anmeldeinformationen im Standardspeicherort gesucht.
   - Wenn Sie nur einen Profilspeicherort angeben, wird bei Ausführung des Befehls das `default`-Profil in dieser Datei mit Anmeldeinformationen gesucht.
   - Wenn Sie sowohl einen Namen als auch einen Speicherort angeben, wird bei Ausführung des Befehls nach dem angegebenen Profil in dieser Datei mit Anmeldeinformationen gesucht.

   Wenn der angegebene Profilname oder Speicherort nicht gefunden wird, löst der Befehl eine Ausnahme aus. Die folgenden Suchschritte werden nur durchgeführt, wenn Sie kein Profil und keinen Speicherort angegeben haben.
3. Mit dem Parameter `-Credential` angegebene Anmeldeinformationen.
4. Das Sitzungsprofil, sofern vorhanden.
5. Das Standardprofil in der folgenden Reihenfolge:
   a. Das Profil `default` im AWS SDK-Speicher.
   b. Das Profil `default` in der Datei mit gemeinsamen AWS-Anmeldeinformationen.
   c. Das Profil `AWS PS Default` im AWS SDK-Speicher.

6. Wenn der Befehl auf einer Amazon-EC2-Instance ausgeführt wird, die für die Verwendung einer IAM-Rolle konfiguriert ist, werden die temporären Anmeldeinformationen der EC2-Instance über das Instance-Profil aufgerufen.

   Weitere Informationen zur Verwendung von IAM-Rollen für Amazon-EC2-Instances finden Sie unter AWS SDK for .NET.

Wenn die angegebenen Anmeldeinformationen im Rahmen dieser Suche nicht gefunden werden, löst der Befehl eine Ausnahme aus.

Verarbeitung von Anmeldeinformationen in AWS Tools for PowerShell Core

Cmdlets in AWS Tools for PowerShell Core akzeptieren AWS-Zugriffs- und geheime Schlüssel oder die Namen von Anmeldeinformationsprofilen, wenn sie ausgeführt werden, ähnlich wie bei AWS Tools for Windows PowerShell. Bei Ausführung unter Windows haben beide Module Zugriff auf die Datei mit AWS SDK for .NET-Anmeldeinformationen (gespeichert in der benutzerspezifischen Datei `AppData\Local\AWSToolkit\RegisteredAccounts.json`).

Diese Datei speichert die Schlüssel verschlüsselt und kann nicht auf einem anderen Computer verwendet werden. Dies ist die erste Datei, die von AWS Tools for PowerShell nach einem Anmeldeinformationsprofil durchsucht wird. In ihr werden auch die Anmeldeprofile von AWS Tools for PowerShell gespeichert. Weitere Informationen zur Datei mit den AWS SDK for .NET-Anmeldeinformationen finden Sie unter Konfigurieren der AWS-Anmeldeinformationen. Das Modul Tools for Windows PowerShell unterstützt derzeit nicht das Speichern von Anmeldeinformationen in anderen Dateien oder an anderen Speicherorten.

Beide Module können die Profile in der Datei mit gemeinsamen AWS-Anmeldeinformationen lesen, die von anderen AWS-SDKs und der AWS CLI verwendet wird. Unter Windows heißt das Standardverzeichnis für diese Datei `C:\Users\<userid>\.aws\credentials`. Unter anderen Betriebssystem wird die Datei im Verzeichnis `~/.aws/credentials` gespeichert. Wenn nicht der Standarddateiname oder -speicherort verwendet wird, kann der betreffende Pfad mit dem Parameter `-ProfileLocation` angegeben werden.

Im SDK-Speicher für Anmeldeinformationen werden die Anmeldeinformationen mithilfe der Kryptografie-APIs von Windows verschlüsselt. Diese APIs sind nicht auf anderen Plattformen verfügbar. Daher verwendet das AWS Tools for PowerShell Core-Modul ausschließlich die Datei mit gemeinsamen AWS-Anmeldeinformationen und unterstützt das Schreiben neuer Anmeldeinformationsprofile in diese Datei.

In den folgenden Beispielskripts mit dem Cmdlet `Set-AWSCredential` werden die Möglichkeiten zur Verarbeitung von Anmeldeinformationsprofilen unter Windows mit AWSPowerShell bzw. AWSPowerShell.NetCore gezeigt.

```
# 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
```

In den folgenden Beispielen wird das Verhalten des Moduls AWSPowerShell.NetCore in den Betriebssystemen Linux und macOS gezeigt.

```
# 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
```

Gemeinsame Anmeldeinformationen in AWS Tools for PowerShell

Die Tools for Windows PowerShell unterstützen die Verwendung der gemeinsamen AWS-Anmeldeinformationen, ähnlich wie die AWS CLI und andere AWS-SDKs. Tools for Windows PowerShell unterstützt jetzt das Lesen und Schreiben von `basic`-, `session`-, und `assume role`-Anmeldeinformationsprofilen in der .NET-Anmeldeinformationsdatei und in der gemeinsamen AWS-Anmeldeinformationsdatei. Diese Funktionalität wird durch einen neuen `Amazon.Runtime.CredentialManagement`-Namespace ermöglicht.

Warning

Um Sicherheitsrisiken zu vermeiden, sollten Sie IAM-Benutzer nicht zur Authentifizierung verwenden, wenn Sie eigens entwickelte Software entwickeln oder mit echten Daten arbeiten. Verwenden Sie stattdessen den Verbund mit einem Identitätsanbieter wie AWS IAM Identity Center.

Note

Die Informationen in diesem Thema beziehen sich auf Situationen, in denen Sie kurz- oder langfristige Anmeldeinformationen manuell abrufen und verwalten müssen. Weitere Informationen zu kurz- und langfristigen Anmeldeinformationen finden Sie unter Andere Authentifizierungsmethoden im Referenzhandbuch für AWS SDKs und Tools.
Als bewährte Sicherheitsmethode verwenden Sie AWS IAM Identity Center, wie unter Tool-Authentifizierung konfigurieren beschrieben.

Die neuen Profiltypen und der Zugriff auf die Datei mit gemeinsamen AWS-Anmeldeinformationen werden über die folgenden Parameter unterstützt, die zu den für Anmeldeinformationen relevanten Cmdlets hinzugefügt wurden: Initialize-AWSDefaultConfiguration, New-AWSCredential und Set-AWSCredential. In Service-Cmdlets können Sie auf Ihre Profile verweisen, indem Sie den allgemeinen Parameter `-ProfileName` hinzufügen.

Verwenden einer IAM-Rolle mit AWS Tools for PowerShell

Die Datei mit gemeinsamen AWS-Anmeldeinformationen ermöglicht zusätzliche Zugriffsarten. Sie können beispielsweise auf Ihre AWS-Ressourcen zugreifen, indem Sie anstelle der langfristigen Anmeldeinformationen eines IAM-Benutzers eine IAM-Rolle verwenden. Dazu müssen Sie über ein Standardprofil verfügen, das über die erforderlichen Berechtigungen verfügt, um die Rolle zu übernehmen. Wenn Sie angeben, dass die AWS Tools for PowerShell ein Profil verwenden sollen, das eine Rolle angegeben hat, suchen die AWS Tools for PowerShell das durch den Parameter `SourceProfile` identifizierte Profil. Diese Anmeldeinformationen werden verwendet, um temporäre Anmeldeinformationen für die durch den Parameter `RoleArn` angegebene Rolle anzufordern. Sie können optional die Verwendung eines Multi-Factor Authentication(MFA)-Geräts oder eines `ExternalId`-Codes verlangen, wenn die Rolle von einem Dritten übernommen wird.

Parametername	Beschreibung
ExternalId	Die benutzerdefinierte externe ID, die zu verwenden ist, wenn eine Rolle übernommen wird, für die dies erforderlich ist. Dies ist in der Regel nur erforderlich, wenn Sie den Zugriff auf Ihr Konto an Dritte delegieren. Der Dritte muss die ExternalID als Parameter angeben, wenn er die zugewiesene Rolle übernimmt. Weitere Informationen finden Sie unter Verwenden einer externen ID, um Dritten Zugriff auf Ihre AWS-Ressourcen zu gewähren im IAM-Benut zerhandbuch.
MfaSerial	Die MFA-Seriennummer, die zu verwenden ist, wenn eine Rolle übernommen wird, für die dies erforderlich ist. Weitere Informationen finden Sie unter Verwenden der Multi-Faktor-Authentifizierung (MFA) in AWS im IAM-Benut zerhandbuch.
RoleArn	Der ARN der übernommenen Rolle für übernommene Rollenanmeldeinformationen. Weitere Informationen zum Erstellen und

Parametername	Beschreibung
	Verwenden von Rollen finden Sie unter IAM-Rollen im IAM-Benutzerhandbuch.
SourceProfile	Der Name des Quellprofils, der von den übernommenen Rollenanmeldeinformationen zu verwenden ist. Die in diesem Profil gefundenen Anmeldeinformationen werden verwendet, um die durch den Parameter `RoleArn` angegebene Rolle zu übernehmen.

Einrichten von Profilen zur Übernahme einer Rolle

Im folgenden Beispiel wird gezeigt, wie ein Quellprofil eingerichtet wird, das die direkte Übernahme einer IAM-Rolle ermöglicht.

Mit dem ersten Befehl wird ein Quellprofil erstellt, auf das mit dem Rollenprofil verwiesen wird. Mit dem zweiten Befehl wird das Rollenprofil erstellt, das die Rolle übernehmen soll. Mit dem dritten Befehl werden die Anmeldeinformationen für das Rollenprofil angezeigt.

```
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
```

Wenn Sie dieses Rollenprofil mit den Tools-for-Windows-Powershell-Service-Cmdlets verwenden möchten, fügen Sie den allgemeinen Parameter `-ProfileName` zum Befehl hinzu, um auf das Rollenprofil zu verweisen. Im folgenden Beispiel wird das im vorherigen Beispiel definierte Rollenprofil verwendet, um auf das Cmdlet Get-S3Bucket zuzugreifen. AWS Tools for PowerShell sucht die Anmeldeinformationen in `my_source_profile`, verwendet diese Anmeldeinformationen, um im

Namen des Benutzers `AssumeRole` aufzurufen, und verwendet dann diese temporären Rollen-Anmeldeinformationen zum Aufrufen von `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
```

Verwenden der Anmeldeinformationsprofil-Typen

Um einen Anmeldeinformationsprofil-Typ festlegen zu können, müssen Sie wissen, welche Parameter die für den Profiltyp erforderlichen Daten bereitstellen.

Anmeldeinformationstyp	Parameter, die Sie verwenden müssen

Basic
 	
Dies sind die langfristigen Anmeldeinformationen für einen IAM-Benutzer
	`-AccessKey`
	
	`-SecretKey`

Sitzung
 	
Dies sind die kurzfristigen Anmeldeinformationen für eine IAM-Rolle, die Sie manuell abrufen, z. B. indem Sie das Cmdlet Use-STSRole direkt aufrufen.
	`-AccessKey`
	
	`-SecretKey`
	
	`-SessionToken`

Rolle:
 	
Dies sind kurzfristige Anmeldeinformationen für eine IAM-Rolle, die die AWS Tools for PowerShell für Sie abrufen.
	`-SourceProfile`
	
	`-RoleArn`
	
	Optional: `-ExternalId`
	
	Optional: `-MfaSerial`

Der allgemeine Parameter **`ProfilesLocation`**

Sie können `-ProfileLocation` verwenden, um die gemeinsame Anmeldeinformationsdatei zu schreiben sowie ein Cmdlet anzuweisen, Daten in der Anmeldeinformationsdatei zu lesen. Durch Hinzufügen des Parameters `-ProfileLocation` wird gesteuert, ob Tools for Windows PowerShell die gemeinsame Anmeldeinformationsdatei oder die .NET-Anmeldeinformationsdatei verwendet. Die folgende Tabelle beschreibt, wie der Parameter in Tools for Windows PowerShell funktioniert.

Profilpositionswert	Profilauflösungsverhalten
Null (nicht festgelegt) oder leer	Durchsuchen Sie zunächst die .NET-Anmeldeinformationsdatei für ein Profil mit dem angegebenen Namen. Wenn das Profil nicht gefunden wird, suchen Sie die Datei mit gemeinsamen AWS-Anmeldeinformationen unter *(user's home directory)* `\.aws\credentials` .
Der Pfad zu einer Datei im Format der Datei mit gemeinsamen AWS-Anmeldeinformationen	Suchen Sie zunächst nur die angegebene Datei für ein Profil mit dem gegebenen Namen.

Speichern der Anmeldeinformationen in einer Anmeldeinformationsdatei

Führen Sie das Cmdlet `Set-AWSCredential` aus, um die Anmeldeinformationen in eine der beiden Anmeldeinformationsdateien zu schreiben und die Datei zu speichern. Das Verfahren wird im folgenden Beispiel gezeigt. Der erste Befehl verwendet `Set-AWSCredential` mit `-ProfileLocation`, um Zugriffs- und geheime Schlüssel zu einem durch den Parameter `-ProfileName` angegebenen Profil hinzuzufügen. Führen Sie in der zweiten Zeile das Cmdlet Get-Content aus, um den Inhalt der Anmeldeinformationsdatei anzuzeigen.

```
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
```

Anzeigen von Anmeldeinformationsprofilen

Führen Sie das Cmdlet Get-AWSCredential aus und fügen Sie den Parameter `-ListProfileDetail` hinzu, um die Typen und Speicherorte der Datei mit Anmeldeinformationen sowie eine Liste der Profilnamen zurückzugeben.

```
PS > Get-AWSCredential -ListProfileDetail

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

Entfernen von Anmeldeinformationsprofilen

Führen Sie zum Entfernen von Anmeldeinformationsprofilen das neue Cmdlet Remove-AWSCredentialProfile aus. Clear-AWSCredential ist veraltet, steht aber weiterhin zur Abwärtskompatibilität zur Verfügung.

Wichtige Hinweise

Nur Initialize-AWSDefaultConfiguration, New-AWSCredential und Set-AWSCredential unterstützen die Parameter für Rollenprofile. Sie können die Rollenparameter nicht direkt in einem Befehl wie beispielsweise `Get-S3Bucket -SourceProfile source_profile_name -RoleArn arn:aws:iam::999999999999:role/role_name` angeben. Dies ist nicht möglich, da Dienst-Cmdlets die Parameter `RoleArn` oder `SourceProfile` nicht direkt unterstützen. Stattdessen müssen Sie diese Parameter in einem Profil speichern und dann den Befehl mit dem Parameter `-ProfileName` aufrufen.

Arbeiten mit AWS-Services in AWS Tools for PowerShell

Dieser Abschnitt zeigt Beispiele dafür, wie Sie AWS Tools for PowerShell für den Zugriff auf AWS-Services verwenden. Anhand dieser Beispiele soll demonstriert werden, wie Sie mithilfe der Cmdlets tatsächliche administrative AWS-Aufgaben ausführen. Diese Beispiele basieren auf Cmdlets, die die Tools für PowerShell bereitstellen. in der AWS Tools for PowerShell-Cmdlet-Referenz finden Sie, welche Cmdlets verfügbar sind.

PowerShell File-Verkettungscodierung

Einige Cmdlets in den AWS Tools for PowerShell bearbeiten vorhandene Dateien oder Datensätze in AWS. Ein Beispiel ist `Edit-R53ResourceRecordSet`, das die API ChangeResourceRecordSets für Amazon Route 53 aufruft.

Wenn Sie Dateien in PowerShell 5.1 oder älteren Versionen bearbeiten oder verketten, codiert PowerShell die Ausgabe in UTF-16 und nicht in UTF-8. Dies kann unerwünschte Zeichen hinzufügen und zu ungültigen Ergebnissen führen. Eine Hexadezimale-Editor kann die unerwünschten Zeichen anzeigen.

Um zu vermeiden, die Dateiausgabe zu UTF-16 konvertieren zu müssen, können Sie Ihren Befehl in das PowerShell-Cmdlet `Out-File` umleiten und UTF-8-Codierung angeben, wie im folgenden Beispiel gezeigt.

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

Wenn Sie AWS CLI-Befehle in der PowerShell-Konsole ausführen, gilt dieselbe Verhaltensweise. Sie können die Ausgabe eines AWS CLI-Befehls zu `Out-File` in der PowerShell-Konsole umleiten. Andere Cmdlets, wie z. B. `Export-Csv` oder `Export-Clixml`, haben auch einen `Encoding`-Parameter. Eine vollständige Liste der Cmdlets mit `Encoding`-Parameter, mit denen Sie die Codierung der Ausgabe einer verketteten Datei korrigieren können, erhalten Sie durch Ausführung des folgenden Befehls.

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

> **Note**
>
> PowerShell 6.0 und neuer, einschließlich PowerShell Core, behält automatisch die UTF-8-Codierung für verkettete Dateiausgaben bei.

Zurückgegebene Objekte für die PowerShell-Tools

Um die AWS Tools for PowerShell in einer nativen PowerShell-Umgebung nützlicher zu machen, ist das von einem AWS Tools for PowerShell-Cmdlet zurückgegebene Objekt ein .NET-Objekt, und nicht das JSON-Textobjekt, das in der Regel von der entsprechenden API im AWS-SDK zurückgegeben wird. So gibt beispielsweise `Get-S3Bucket` eine `Buckets`-Sammlung und kein Amazon-S3-JSON-Antwortobjekt aus. Die `Buckets`-Sammlung kann in der PowerShell-Pipeline platziert werden, um geeignete Interaktionen zu ermöglichen. Entsprechend gibt `Get-EC2Instance` eine `Reservation`-.NET-Objektsammlung und kein `DescribeEC2Instances`-JSON-Ergebnisobjekt aus. Dieses Verhalten ist gewollt und ermöglicht eine konsistentere AWS Tools for PowerShell-Umgebung mit der idiomatischen PowerShell.

Die tatsächlichen Service-Antworten stehen Ihnen zur Verfügung, wenn Sie sie benötigen. Sie werden als `note`-Eigenschaften auf den zurückgegebenen Objekten gespeichert. Für API-Aktionen, die das Paging über `NextToken`-Felder unterstützen, werden sie außerdem als `note`-Eigenschaften angefügt.

Amazon EC2

In diesem Abschnitt werden die erforderlichen Schritte zum Starten einer Amazon-EC2-Instance beschrieben, z. B.:

- Abrufen einer Liste von Amazon Machine Images (AMIs)
- Erstellen Sie ein Schlüsselpaar für die SSH-Authentifizierung.
- Erstellen und konfigurieren Sie eine Amazon-EC2-Sicherheitsgruppe.
- Starten der Instance und Abrufen der Instance-Informationen

Amazon S3

Der Abschnitt führt Sie durch die erforderlichen Schritte zum Erstellen einer statischen Website, die in Amazon S3 gehostet wird. Er zeigt Folgendes:

- Erstellen und Löschen von Amazon-S3-Buckets.
- Hochladen von Dateien als Objekte in einen Amazon-S3-Bucket.
- Löschen von Objekten aus einem Amazon-S3-Bucket.
- Aktivieren eines Amazon-S3-Buckets als Website.

AWS Lambda und AWS Tools for PowerShell

Dieser Abschnitt enthält eine kurze Übersicht über das AWS-Lambda-Tools-for-PowerShell-Modul und beschreibt die erforderlichen Schritte zur Einrichtung des Moduls.

Amazon SNS und Amazon SQS

In diesem Abschnitt werden die Schritte beschrieben, die zum Abonnieren einer Amazon-SQS-Warteschlange für ein Amazon-SNS-Thema erforderlich sind. Er zeigt Folgendes:

- Erstellen Sie ein Amazon-SNS-Thema.
- Erstellen einer Amazon SQS-Warteschlange
- Abonnieren der -Warteschlange für das -Thema
- Senden einer Mitteilung an das Thema
- Empfangen einer Mitteilung aus der Warteschlange

CloudWatch

In diesem Abschnitt finden Sie ein Beispiel zum Veröffentlichen benutzerdefinierter Daten in CloudWatch.

- Veröffentlichen einer benutzerdefinierten Kennzahl im CloudWatch-Dashboard.

Weitere Informationen finden Sie unter:

- Erste Schritte mit der AWS Tools for Windows PowerShell

Themen

Amazon S3 und Tools for Windows PowerShell

In diesem Abschnitt erstellen wir über AWS Tools for Windows PowerShell unter Verwendung von Amazon S3 und CloudFront eine statische Website. Während des Vorgangs werden verschiedene gebräuchliche Aufgaben mit diesen Services gezeigt. Diese Schritte orientieren sich am Handbuch Erste Schritte für das Hosten einer statischen Website, in der ein ähnlicher Prozess unter Verwendung der AWS-Managementkonsole beschrieben wird.

Die hier gezeigten Befehle setzen voraus, dass Sie Standardanmeldeinformationen und eine Standardregion für die PowerShell-Sitzung festgelegt haben. Daher werden beim Aufrufen der Cmdlets keine Anmeldeinformationen und Regionen übergeben.

> **Note**
>
> Da es derzeit keine Amazon-S3-API zum Umbenennen von Buckets oder Objekten gibt, steht kein einzelnes Tools-for-Windows-PowerShell-Cmdlet für diese Aufgabe zur Verfügung. Zum Umbenennen von Objekten in S3 empfehlen wir, das Objekt zu kopieren und neu zu benennen. Führen Sie dazu das Cmdlet Copy-S3Object aus und löschen Sie das ursprüngliche Objekt mit dem Cmdlet Remove-S3Object.

Weitere Informationen finden Sie auch unter

- Arbeiten mit AWS-Services in AWS Tools for PowerShell
- Hosten einer statischen Website auf Amazon S3
- Amazon S3-Konsole

Themen

Erstellen eines Amazon-S3-Buckets, Verifizieren der Region und (optional) Entfernen des Buckets

Mit dem `New-S3Bucket`-Cmdlet können Sie einen neuen Amazon-S3-Bucket erstellen. In den folgenden Beispielen wird ein Bucket mit dem Namen erstellt `website-example`. Der Name des Buckets muss regionsübergreifend eindeutig sein. Im Beispiel wird der Bucket in der Region `us-west-1` erstellt.

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

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

Die Region, in der sich der Bucket befindet, kann mit dem Cmdlet `Get-S3BucketLocation` ermittelt werden.

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

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

Wenn Sie dieses Tutorial abgeschlossen haben, können Sie diesen Bucket mithilfe der folgenden Zeile entfernen. Wir empfehlen aber, den Bucket nicht zu entfernen, da er in späteren Beispielen verwendet wird.

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

Beachten Sie, dass das Entfernen eines Buckets einige Zeit in Anspruch nehmen kann. Wenn Sie versuchen, sofort einen Bucket mit dem gleichen Namen zu erstellen, kann das Cmdlet `New-S3Bucket` fehlschlagen, bis der alte Bucket vollständig verschwunden ist.

Weitere Informationen finden Sie unter:

- Arbeiten mit AWS-Services in AWS Tools for PowerShell
- PUT Bucket (Amazon S3-Servicereferenz)
- AWS-PowerShell-Regionen für Amazon S3

Konfigurieren eines Amazon-S3-Buckets als Website und Aktivieren der Protokollierung

Verwenden Sie das Cmdlet `Write-S3BucketWebsite`, um einen Amazon-S3-Bucket als statische Website zu konfigurieren. Das folgende Beispiel gibt einen Namen der `index.html`-Datei für die Standardwebseite mit dem Inhalt und einen Namen für die `error.html`-Datei für die Standardwebseite mit den Fehlern an. Beachten Sie, dass dieses Cmdlet diese Seiten nicht erstellt. Sie müssen als Amazon S3-Objekte hochgeladen werden.

```
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    :
```

Weitere Informationen finden Sie unter:

- Arbeiten mit AWS-Services in AWS Tools for PowerShell
- Put Bucket Website (Amazon S3-API-Referenz)
- Put Bucket ACL (Amazon S3-API-Referenz)

Hochladen von Objekten in einen Amazon-S3-Bucket

Verwenden Sie das Cmdlet `Write-S3Object` zum Hochladen von Dateien (als Objekte) aus dem lokalen Dateisystem in einen Amazon-S3-Bucket. Das folgende Beispiel erstellt zwei einfache HTML-Dateien, lädt sie in einen Amazon-S3-Bucket hoch und verifiziert die hochgeladenen Objekte. Der Parameter `-File` von `Write-S3Object` gibt den Namen der Datei im lokalen Dateisystem an. Der Parameter `-Key` gibt den Namen an, den das entsprechende Objekt in Amazon S3 hat.

Amazon leitet "content-type" der Objekte aus den Dateinamenserweiterungen – in diesem Fall ".html" – ab.

```
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
```

Vordefinierte ACL-Optionen

Die Werte zum Angeben vordefinierter ACLs mit Tools for Windows PowerShell entsprechen denen, die von verwendet werden AWS SDK for .NET. Beachten Sie jedoch, dass diese Werte sich von den Werten unterscheiden, die die Amazon-S3`Put Object`-Aktion verwendet. Die Tools for Windows PowerShell unterstützen die folgenden vorgefertigen ACLs:

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

Weitere Informationen zu diesen vordefinierten ACL-Einstellungen finden Sie unter Zugriffskontrolllisten (ACL) – Übersicht.

Hinweis zu mehrteiligen Uploads

Wenn Sie die Amazon-S3-API in eine Datei mit mehr als 5 GB hochladen, müssen Sie den mehrteiligen Upload nutzen. Das von Tools for Windows PowerShell bereitgestellte Cmdlet `Write-S3Object` kann Datei-Uploads von mehr als 5 GB transparent verarbeiten.

Testen der Website

Nun können Sie die Website testen, indem Sie sie mit einem Browser aufrufen. URLs für in Amazon S3 gehostete statische Websites weisen ein Standardformat auf.

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

Beispiel:

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

Weitere Informationen finden Sie unter:

- Arbeiten mit AWS-Services in AWS Tools for PowerShell
- Put Object (Amazon S3-API-Referenz)
- Vordefinierte ACLs (Amazon S3-API-Referenz)

Löschen von Amazon-S3-Objekten und -Buckets

In diesem Abschnitt wird beschrieben, wie Sie die in den vorhergehenden Abschnitten erstellte Website löschen. Sie können einfach die Objekte für die HTML-Dateien und danach den Amazon-S3-Bucket für die Website löschen.

Führen Sie das Cmdlet `Remove-S3Object` aus, um die Objekte für die HTML-Dateien im Amazon-S3-Bucket zu löschen.

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

Der Rückgabewert `False` ist ein Artefakt, das sich aus der Art der Anforderungsverarbeitung durch Amazon S3 ergibt. In diesem Kontext weist der Wert nicht auf ein Problem hin.

Jetzt können Sie das Cmdlet `Remove-S3Bucket` ausführen, um den nun leeren Amazon-S3-Bucket für die Website zu löschen.

```
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    :
```

In AWS Tools for PowerShell-Versionen ab 1.1 können Sie den Parameter `-DeleteBucketContent` zu `Remove-S3Bucket` hinzufügen. Dadurch werden vor dem Löschen des Buckets zunächst alle Objekte und Objektversionen aus diesem entfernt. Je nach Anzahl der Objekte oder Objektversionen im Bucket kann dieser Vorgang längere Zeit dauern. In den Versionen von Tools for Windows PowerShell vor 1.1 konnten nur leere Buckets mit `Remove-S3Bucket` gelöscht werden.

> **Note**
>
> Wenn Sie den Parameter `-Force` nicht hinzufügen, fordern die AWS Tools for PowerShell Sie zur Bestätigung auf, bevor das Cmdlet ausgeführt wird.

Weitere Informationen finden Sie unter:

- Arbeiten mit AWS-Services in AWS Tools for PowerShell
- Delete Object (Amazon S3-API-Referenz)
- DeleteBucket (Amazon S3-API-Referenz)

Hochladen von Inline-Textinhalt nach Amazon S3

Das Cmdlet `Write-S3Object` unterstützt das Hochladen von Inline-Textinhalt nach Amazon S3. Sie können mit dem Parameter `-Content` (Alias `-Text`) Textinhalt nach Amazon S3 hochladen, ohne dass dieser zuerst in eine Datei eingefügt werden muss. Der Parameter akzeptiert sowohl einfache einzeilige Zeichenfolgen als auch – wie hier gezeigt – Zeichenfolgen mit mehreren Zeilen.

```
PS > # Specifying content in-line, single line text:
PS > write-s3object mybucket -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 mybucket -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 mybucket -key myobject.txt -content $x
```

Amazon EC2 und Tools for Windows PowerShell

Sie können typische Aufgaben im Zusammenhang mit Amazon EC2 unter Verwendung von AWS Tools for PowerShell ausführen.

Die hier gezeigten Beispielbefehle setzen voraus, dass Sie Standardanmeldeinformationen und eine Standardregion für die PowerShell-Sitzung festgelegt haben. Aus diesem Grund geben wir im Aufruf von Cmdlets weder Anmeldeinformationen noch eine Region an. Weitere Informationen finden Sie unter Erste Schritte mit der AWS Tools for Windows PowerShell.

Themen

- Erstellen eines Schlüsselpaars
- Erstellen Sie eine Sicherheitsgruppe mit Windows PowerShell
- Suchen eines Amazon Machine Images mit Windows PowerShell
- Starten Sie eine Amazon EC2 EC2-Instance mit Windows PowerShell

Erstellen eines Schlüsselpaars

Im folgenden Beispiel für `New-EC2KeyPair` wird ein Schlüsselpaar erstellt und in der PowerShell-Variablen `$myPSKeyPair` gespeichert.

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

Übergeben Sie das Schlüsselpaarobjekt mit einem Pipe-Zeichen an das Cmdlet `Get-Member`, um die Struktur des Objekts anzuzeigen.

```
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;}
```

Übergeben Sie das Schlüsselpaarobjekt mit einem Pipe-Zeichen an das Cmdlet `Format-List`, um die Werte der Elemente `KeyName`, `KeyFingerprint` und `KeyMaterial` anzuzeigen. (Die Ausgabe wurde zur besseren Lesbarkeit gekürzt.)

```
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-----
```

Der private Schlüssel des Schlüsselpaares wird im Element `KeyMaterial` gespeichert. Der öffentliche Schlüssel wird in gespeichert AWS. Der öffentliche Schlüssel kann zwar nicht aus AWS abgerufen werden, Sie können ihn aber durch Vergleichen des `KeyFingerprint`-Werts für den privaten Schlüssel mit dem von AWS für den öffentlichen Schlüssel zurückgegebenen Wert verifizieren.

Anzeigen des Schlüsselpaar-Fingerabdrucks

Mit dem Cmdlet `Get-EC2KeyPair` können Sie den Schlüsselpaar-Fingerabdruck anzeigen.

```
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
```

Speichern des privaten Schlüssels

Um den privaten Schlüssel in einer Datei zu speichern, übergeben Sie das Element `KeyFingerMaterial` mit einem Pipe-Zeichen an das Cmdlet `Out-File`.

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

Sie müssen beim Schreiben des privaten Schlüssels in eine Datei den Parameterwert `-Encoding ascii` angeben. Andernfalls können Tools wie beispielsweise `openssl` die Datei möglicherweise nicht richtig lesen. Das Format der beschriebenen Datei kann mit einem Befehl wie dem folgenden geprüft werden:

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

(Das Tool `openssl` ist nicht in AWS Tools for PowerShell oder dem AWS SDK for .NET enthalten.)

Entfernen des Schlüsselpaars

Sie benötigen das Schlüsselpaar, um eine Instance zu starten und eine Verbindung damit herzustellen. Wenn Sie ein Schlüsselpaar nicht mehr benötigen, können Sie es entfernen. Mit dem CmdletRemove-EC2KeyPair können Sie den öffentlichen Schlüssel aus AWS entfernen. Drücken Sie bei Aufforderung die Taste `Enter`, um das Schlüsselpaar zu entfernen.

```
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"):
```

Die Variable `$myPSKeyPair` ist weiterhin in der aktuellen PowerShell-Sitzung definiert und enthält die Schlüsselpaarinformationen. Außerdem ist die Datei `myPSKeyPair.pem` vorhanden. Allerdings ist der private Schlüssel nicht mehr gültig, da der öffentliche Schlüssel des Schlüsselpaares nicht mehr in gespeichert ist AWS.

Erstellen Sie eine Sicherheitsgruppe mit Windows PowerShell

Sie können den verwenden AWS Tools for PowerShell , um eine Sicherheitsgruppe zu erstellen und zu konfigurieren. Wenn Sie eine Sicherheitsgruppe erstellen, können Sie angeben, ob sie für EC2-Classic oder EC2-VPC bestimmt ist. Die Antwort ist die ID der Sicherheitsgruppe.

Wenn Sie eine Verbindung zur Instance herstellen müssen, muss die Sicherheitsgruppe so konfiguriert werden, dass SSH-Datenverkehr (Linux) oder RDP-Datenverkehr (Windows) möglich ist.

Themen

Voraussetzungen

Sie benötigen die öffentliche IP-Adresse des Computers in CIDR-Schreibweise. Sie können einen Service verwenden, um die öffentliche IP-Adresse des Computers abzufragen. Beispielsweise stellt Amazon folgenden Service bereit: http://checkip.amazonaws.com/ oder https://checkip.amazonaws.com/. Geben Sie zum Auffinden eines anderen Service, der Ihnen Ihre IP-Adresse nennt, den Suchausdruck "wie ist meine IP-Adresse" ein. Wenn Sie eine Verbindung über einen Internetdienstanbieter oder hinter Ihrer Firewall ohne statische IP-Adresse herstellen, müssen Sie den IP-Adressbereich ermitteln, der von Ihren Clientcomputern verwendet werden kann.

> **Warning**
>
> Wenn Sie `0.0.0.0/0` angeben, aktivieren Sie den Datenverkehr von allen IP-Adressen auf der ganzen Welt. Dieser Ansatz ist möglicherweise für SSH- und RDP-Protokolle in einer Testumgebung für kurze Zeit zulässig, aber für Produktionsumgebungen sehr unsicher. Achten Sie in Produktionsumgebungen darauf, nur Zugriff von der entsprechenden individuellen IP-Adresse oder vom entsprechenden Adressbereich zu autorisieren.

Erstellen einer Sicherheitsgruppe für EC2-Classic

> **Warning**
>
> EC2-Classic wird am 15. August 2022 eingestellt. Wir empfehlen Ihnen die Migration von EC2-Classic zu einer VPC. Weitere Informationen finden Sie unter Migration von EC2-Classic zu einer VPC im Amazon EC2 EC2-Benutzerhandbuch oder im Amazon EC2 EC2-Benutzerhandbuch. Lesen Sie außerdem den Blogbeitrag EC2-Classic Networking is Retiring – Here's How to Prepare (EC2-Classic Networking wird außer Betrieb genommen – so bereiten Sie sich vor).

Im folgenden Beispiel wird mit dem Cmdlet `New-EC2SecurityGroup` eine Sicherheitsgruppe für `EC2-Classic` erstellt.

```
PS > New-EC2SecurityGroup -GroupName myPSSecurityGroup -GroupDescription "EC2-Classic
 from PowerShell"

sg-0a346530123456789
```

Verwenden Sie zum Anzeigen der anfänglichen Konfiguration das Cmdlet `Get-EC2SecurityGroup`.

```
PS > Get-EC2SecurityGroup -GroupNames myPSSecurityGroup

Description         : EC2-Classic from PowerShell
GroupId             : sg-0a346530123456789
GroupName           : myPSSecurityGroup
IpPermissions       : {}
IpPermissionsEgress : {Amazon.EC2.Model.IpPermission}
```

```
OwnerId             : 123456789012
Tags                : {}
VpcId               : vpc-9668ddef
```

Um die Sicherheitsgruppe so zu konfigurieren, dass eingehender Datenverkehr über TCP-Port 22 (SSH) und TCP-Port 3389 zulässig ist, verwenden Sie das Cmdlet `Grant-EC2SecurityGroupIngress`. Das folgende Beispielskript zeigt, wie Sie SSH-Datenverkehr von einer einzelnen IP-Adresse, `203.0.113.25/32`, zuzulassen könnten.

```
$cidrBlocks = New-Object 'collections.generic.list[string]'
$cidrBlocks.add("203.0.113.25/32")
$ipPermissions = New-Object Amazon.EC2.Model.IpPermission
$ipPermissions.IpProtocol = "tcp"
$ipPermissions.FromPort = 22
$ipPermissions.ToPort = 22
ipPermissions.IpRanges = $cidrBlocks
Grant-EC2SecurityGroupIngress -GroupName myPSSecurityGroup -IpPermissions
 $ipPermissions
```

Führen Sie das Cmdlet `Get-EC2SecurityGroup` erneut aus, um zu verifizieren, dass die Sicherheitsgruppe aktualisiert worden ist. Beachten Sie, dass Sie für `EC2-Classic` keine Regel für ausgehenden Datenverkehr angeben können.

```
PS > Get-EC2SecurityGroup -GroupNames myPSSecurityGroup

OwnerId             : 123456789012
GroupName           : myPSSecurityGroup
GroupId             : sg-0a346530123456789
Description         : EC2-Classic from PowerShell
IpPermissions       : {Amazon.EC2.Model.IpPermission}
IpPermissionsEgress : {}
VpcId               :
Tags                : {}
```

Verwenden Sie zum Anzeigen der Sicherheitsgruppenregel die Eigenschaft `IpPermissions`.

```
PS > (Get-EC2SecurityGroup -GroupNames myPSSecurityGroup).IpPermissions

IpProtocol       : tcp
FromPort         : 22
ToPort           : 22
```

```
UserIdGroupPairs : {}
IpRanges         : {203.0.113.25/32}
```

Erstellen einer Sicherheitsgruppe für EC2-VPC

Im folgenden Beispiel für `New-EC2SecurityGroup` wird der Parameter `-VpcId` hinzugefügt, um eine Sicherheitsgruppe für die angegebene VPC zu erstellen.

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

Verwenden Sie zum Anzeigen der anfänglichen Konfiguration das Cmdlet `Get-EC2SecurityGroup`. Standardmäßig enthält die Sicherheitsgruppe für eine VPC eine Regel, die den gesamten ausgehenden Datenverkehr zulässt. Beachten Sie, dass Sie eine Sicherheitsgruppe für EC2-VPC nicht über den Namen referenzieren können.

```
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                : {}
```

Verwenden Sie das Cmdlet `New-Object`, um die Berechtigungen für eingehenden Datenverkehr auf TCP-Port 22 (SSH) und TCP-Port 3389 zu definieren. Im folgenden Beispielskript werden Berechtigungen für TCP-Port 22 und 3389 von einer einzelnen IP-Adresse, `203.0.113.25/32`, definiert.

```
$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 )
```

Verwenden Sie das Cmdlet `Get-EC2SecurityGroup` erneut, um zu verifizieren, dass die Sicherheitsgruppe aktualisiert wurde.

```
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                : {}
```

Um die eingehenden Regeln anzuzeigen, können Sie die Eigenschaft `IpPermissions` aus dem vom vorherigen Befehl zurückgegebenen Sammlungsobjekt abrufen.

```
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}
```

Suchen eines Amazon Machine Images mit Windows PowerShell

Wenn Sie eine Amazon-EC2-Instance starten, müssen Sie ein Amazon Machine Image (AMI) angeben, das als Vorlage für die Instance dient. Die IDs für die AWS-Windows-AMIs ändern

sich jedoch häufig, weil AWS mit den Updates und Sicherheitsverbesserungen auch neue AMIs bereitstellt. Sie können die Cmdlets Get-EC2Image und Get-EC2ImageByName verwenden, um die aktuellen Windows-AMIs zu suchen und ihre IDs abzurufen.

Themen

Get-EC2Image

Das Cmdlet `Get-EC2Image` ruft eine Liste der AMIs ab, die Sie verwenden können.

Verwenden Sie den Parameter `-Owner` mit dem Array-Wert `amazon, self`, damit `Get-EC2Image` nur AMIs abruft, die Amazon oder Ihnen gehören. In diesem Zusammenhang bezeichnet Sie den Benutzer, dessen Anmeldeinformationen Sie zum Aufrufen des Cmdlets verwendet haben.

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

Sie können den Umfang der Ergebnisse mit dem Parameter `-Filter` definieren. Um den Filter anzugeben, erstellen Sie ein Objekt des Typs `Amazon.EC2.Model.Filter`. Verwenden Sie beispielsweise den folgenden Filter, um nur Windows-AMIs anzuzeigen.

```
$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
```

Es folgt ein Beispiel für ein vom Cmdlet zurückgegebenes AMI. Die tatsächliche Ausgabe des vorherigen Befehls stellt Informationen für viele AMIs bereit.

```
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

Mit dem Cmdlet `Get-EC2ImageByName` können Sie die Liste der AWS-Windows-AMIs basierend auf dem Typ der für Sie relevanten Serverkonfiguration filtern.

Bei Ausführung ohne Parameter (siehe unten) gibt das Cmdlet den vollständigen Satz der aktuellen Filternamen aus:

```
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
```

Um die zurückgegebenen Images zu beschränken, geben Sie mindestens einen Filternamen mit dem Parameter `Names` an.

```
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
```

Starten Sie eine Amazon EC2 EC2-Instance mit Windows PowerShell

Um eine Amazon EC2-Instance zu starten, benötigen Sie das Schlüsselpaar und die von Ihnen zuvor erstellte Sicherheitsgruppe. Sie benötigen außerdem die ID eines Amazon Machine Images (AMI). Weitere Informationen finden Sie in der folgenden -Dokumentation:

- Erstellen eines Schlüsselpaars
- Erstellen Sie eine Sicherheitsgruppe mit Windows PowerShell
- Suchen Sie mithilfe von Windows nach einem Amazon-Maschinen-Image PowerShell

> **Important**
>
> Wenn Sie eine Instance starten, die nicht im kostenlosen Kontingent enthalten ist, fallen für diese Instance nach dem Start Gebühren an. Ihnen wird die Zeit berechnet, die die Instance ausgeführt wird, auch wenn diese nicht genutzt wird.

Themen

- Starten einer Instance in EC2-Classic
- Starten einer Instance in einer VPC
- Starten einer Spot-Instance in einer VPC

Starten einer Instance in EC2-Classic

> **Warning**
>
> EC2-Classic wird am 15. August 2022 eingestellt. Wir empfehlen Ihnen die Migration von EC2-Classic zu einer VPC. Weitere Informationen finden Sie unter Migration von EC2-Classic zu einer VPC im Amazon EC2 EC2-Benutzerhandbuch oder im Amazon EC2 EC2-Benutzerhandbuch. Lesen Sie außerdem den Blogbeitrag EC2-Classic Networking is Retiring – Here's How to Prepare (EC2-Classic Networking wird außer Betrieb genommen – so bereiten Sie sich vor).

Der folgende Befehl erstellt und startet eine einzelne `t1.micro`-Instance.

```
PS > New-EC2Instance -ImageId ami-c49c0dac `
    -MinCount 1 `
    -MaxCount 1 `
    -KeyName myPSKeyPair `
    -SecurityGroups myPSSecurityGroup `
    -InstanceType t1.micro

ReservationId   : r-b70a0ef1
OwnerId         : 123456789012
RequesterId     :
Groups          : {myPSSecurityGroup}
GroupName       : {myPSSecurityGroup}
Instances       : {}
```

Die Instance weist anfangs den Status `pending` auf, wechselt aber nach wenigen Minuten in den Status `running`. Verwenden Sie zum Anzeigen von Informationen zur Instance das Cmdlet `Get-EC2Instance`. Wenn mehr als eine Instance vorliegt, können Sie die Ergebnisse mit dem Parameter `Filter` über die Reservierungs-ID filtern. Erstellen Sie zunächst ein Objekt des Typs `Amazon.EC2.Model.Filter`. Rufen Sie dann das Cmdlet `Get-EC2Instance`, das den Filter verwendet, und zeigen Sie dann die Eigenschaft `Instances` an.

```
PS > $reservation = New-Object 'collections.generic.list[string]'
PS > $reservation.add("r-5caa4371")
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          : t1.micro
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              :
Tags                  : {}
VirtualizationType    : hvm
VpcId                 :
```

Starten einer Instance in einer VPC

Mit dem folgenden Befehl wird eine einzelne `m1.small`-Instance im angegebenen privaten Subnetz gestartet. Die Sicherheitsgruppe muss für das angegebene Subnetz gültig sein.

```
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       : {}
```

Die Instance weist anfangs den Status `pending` auf, wechselt aber nach wenigen Minuten in den Status `running`. Verwenden Sie zum Anzeigen von Informationen zur Instance das Cmdlet `Get-EC2Instance`. Wenn mehr als eine Instance vorliegt, können Sie die Ergebnisse mit dem Parameter `Filter` über die Reservierungs-ID filtern. Erstellen Sie zunächst ein Objekt des Typs `Amazon.EC2.Model.Filter`. Rufen Sie dann das Cmdlet `Get-EC2Instance`, das den Filter verwendet, und zeigen Sie dann die Eigenschaft `Instances` an.

```
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
```

Starten einer Spot-Instance in einer VPC

Im folgenden Beispielskript wird eine Spot-Instance im angegebenen Subnetz angefordert. Die Sicherheitsgruppe muss für die VPC mit dem angegebenen Subnetz erstellt worden sein.

```
$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 und AWS Tools for PowerShell

Mit dem Modul AWSLambdaPSCore können Sie mithilfe der .NET Core 2.1-Laufzeit AWS Lambda-Funktionen in PowerShell Core 6.0 entwickeln. PowerShell-Entwickler können anhand Lambda in der PowerShell-Umgebung AWS-Ressourcen verwaltet und Automatisierungsskripts schreiben. Mit PowerShell-Support in Lambda können Sie PowerShell-Skripts oder Funktionen als Reaktion auf ein Ereignis, wie z. B. ein Amazon-S3-Ereignis oder ein geplantes Amazon-CloudWatch-Ereignis, ausführen. Das Modul AWSLambdaPSCore ist ein separates AWS-Modul für PowerShell und nicht Bestandteil der AWS Tools for PowerShell. Zudem führt die Installation des Moduls AWSLambdaPSCore nicht automatisch zur Installation der AWS Tools for PowerShell.

Nach der Installation des AWSLambDAPscore-Moduls können Sie alle verfügbaren PowerShell-Cmdlets verwenden oder eigene Funktionen entwickeln, um Serverless-Funktionen zu erstellen. Das AWS-Lambda-Tools-for-PowerShell-Modul enthält Projektvorlagen für PowerShell-basierte Serverless-Anwendungen sowie Tools, mit denen Sie Ihre Projekte in AWS veröffentlichen können.

Der AWSLambdaPSCore-Modul-Support ist in allen Regionen verfügbar, in denen auch Lambda unterstützt wird. Weitere Informationen zu den unterstützen Regionen finden Sie in der Tabelle der AWS-Regionen.

Voraussetzungen

Die folgenden Schritte sind erforderlich, bevor Sie das AWSLambdaPSCore-Modul installieren und verwenden können. Weitere Informationen zu diesen Schritten finden Sie unter dem Abschnitt zum Einrichten einer PowerShell-Entwicklungsumgebung im AWS Lambda-Entwicklerhandbuch.

- Installieren Sie die richtige Version von PowerShell – Der Lambda-Support für PowerShell basiert auf der plattformübergreifenden PowerShell Core 6.0-Version. Sie können PowerShell-Lambda-Funktionen unter Windows, Linux oder Mac entwickeln. Wenn Sie nicht mindestens diese Version von PowerShell installiert haben, finden Sie die entsprechenden Anweisungen auf der Dokumentationswebsite für Microsoft PowerShell.
- Installieren Sie das .NET Core 2.1 SDK – Da PowerShell Core auf.NET Core basiert, verwendet der Lambda-Support für PowerShell die gleiche .NET-Core-2.1-Lambda-Laufzeit für sowohl .NET-Core- als auch PowerShell-Lambda-Funktionen. Die Lambda-PowerShell-Cmdlets zur Veröffentlichung verwenden das .NET Core 2.1 SDK, um das Lambda-Bereitstellungspaket zu erstellen. Das .NET Core 2.1 SDK finden Sie im Microsoft Download Center. Achten Sie darauf, das SDK und nicht die Laufzeit zu installieren.

Installieren Sie das AWSLambdaPSCore-Modul.

Wenn die Voraussetzungen erfüllt sind, können Sie das AWSLambdaPSCore-Modul installieren. Führen Sie den folgenden Befehl in einer PowerShell-Sitzung aus.

```
PS> Install-Module AWSLambdaPSCore -Scope CurrentUser
```

Sie können jetzt mit der Entwicklung von Lambda-Funktionen in PowerShell beginnen. Weitere Informationen zu den ersten Schritten finden Sie im Abschnitt zum Programmiermodell für die Erstellung von Lambda-Funktionen in PowerShell im AWS Lambda-Entwicklerhandbuch.

Weitere Informationen finden Sie unter:

- Ankündigung von Lambda-Support für PowerShell Core auf dem AWS-Entwickler-Blog
- AWSLambdaPSCore-Modul auf der PowerShell Gallery-Website
- Einrichten einer PowerShell-Entwicklungsumgebung
- AWS-Lambda Tools für PowerShell auf GitHub
- AWS-Lambda-Konsole

Amazon SQS, Amazon SNS und Tools for Windows PowerShell

Dieser Abschnitt enthält Beispiele, die Folgendes veranschaulichen:

- Erstellen einer Amazon-SQS-Warteschlange und Abrufen des Amazon-Ressourcennamens (ARN).
- Erstellen Sie ein Amazon SNS-Thema.
- Erteilen Sie Berechtigungen für das SNS-Thema, damit es Nachrichten an die Warteschlange senden kann.
- Abonnieren der Warteschlange für das SNS-Thema
- Gewähren Sie IAM-Benutzern oder AWS-Konten Berechtigungen zur Veröffentlichung im SNS-Thema und zum Lesen von Nachrichten in der SQS-Warteschlange.
- Verifizieren Sie die Ergebnisse, indem Sie eine Nachricht im Thema veröffentlichen und die Nachricht in der Warteschlange lesen.

Erstellen einer Amazon-SQS-Warteschlange und Abrufen des Warteschlangen-ARN

Mit dem folgenden Befehl wird eine SQS-Warteschlange in Ihrer Standardregion erstellt. Die Ausgabe zeigt die URL der neuen Warteschlange an.

```
PS > New-SQSQueue -QueueName myQueue
https://sqs.us-west-2.amazonaws.com/123456789012/myQueue
```

Der folgende Befehl ruft den ARN der Warteschlange ab.

```
PS > Get-SQSQueueAttribute -QueueUrl https://sqs.us-west-2.amazonaws.com/123456789012/
myQueue -AttributeName QueueArn
...
QueueARN               : arn:aws:sqs:us-west-2:123456789012:myQueue
...
```

Erstellen Sie ein Amazon SNS-Thema.

Mit dem folgenden Befehl wird ein SNS-Thema in Ihrer Standardregion erstellt und der ARN des neuen Themas wird zurückgegeben.

```
PS > New-SNSTopic -Name myTopic
arn:aws:sns:us-west-2:123456789012:myTopic
```

Gewähren von Berechtigungen für das SNS-Thema

Das folgende Beispielskript erstellt sowohl eine SQS-Warteschlange als auch ein SNS-Thema und erteilt Berechtigungen für das SNS-Thema, damit es Nachrichten an die SQS-Warteschlange senden kann:

```
# 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 }
```

Abonnieren der Warteschlange für das SNS-Thema

Mit dem folgenden Befehl wird die Warteschlange `myQueue` für das SNS-Thema `myTopic` abonniert und die Abonnement-ID zurückgegeben:

```
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
```

Gewähren von Berechtigungen

Mit dem folgenden Befehl wird die Berechtigung zum Ausführen der Aktion `sns:Publish` für das Thema `myTopic` gewährt

```
PS > Add-SNSPermission `
    -TopicArn arn:aws:sns:us-west-2:123456789012:myTopic `
    -Label ps-cmdlet-topic `
    -AWSAccountIds 123456789012 `
    -ActionNames publish
```

Mit dem folgenden Befehl wird die Berechtigung zum Ausführen der Aktionen `sqs:ReceiveMessage` und `sqs:DeleteMessage` für die Warteschlange `myQueue` gewährt:

```
PS > Add-SQSPermission `
    -QueueUrl https://sqs.us-west-2.amazonaws.com/123456789012/myQueue `
    -AWSAccountId "123456789012" `
    -Label queue-permission `
    -ActionName SendMessage, ReceiveMessage
```

Überprüfen der Ergebnisse

Mit dem folgenden Befehl werden Ihre neue Warteschlange und das neue Thema getestet, indem eine Nachricht im SNS-Thema `myTopic` veröffentlicht und die `MessageId` zurückgegeben wird.

```
PS > Publish-SNSMessage `
    -TopicArn arn:aws:sns:us-west-2:123456789012:myTopic `
    -Message "Have A Nice Day!"
728180b6-f62b-49d5-b4d3-3824bb2e77f4
```

Mit dem folgenden Befehl wird die Nachricht aus der SQS-Warteschlange `myQueue` zurückgegeben und angezeigt:

```
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
+qTE2lId2RPkFOeGtLGawTsSPTWEvJdDbLlf7E0zZ0q1niXTUtpsZ8Swx01X3QO6u9i9qBFt0ekJFZNJp6Avu05hIklb4yc
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 aus AWS Tools for Windows PowerShell

Dieser Abschnitt zeigt ein Beispiel zum Veröffentlichen benutzerdefinierter Kennzahlen in Tools for Windows PowerShell unter Verwendung von CloudWatch.

Dieses Beispiel setzt voraus, dass Sie Standardanmeldeinformationen und eine Standardregion für die PowerShell-Sitzung festgelegt haben.

Veröffentlichen einer benutzerdefinierten Kennzahl im CloudWatch-Dashboard

Der folgende PowerShell-Code initialisiert ein CloudWatch-`MetricDatum`-Objekt und sendet es an den Service. Sie können das Ergebnis dieser Operation anzeigen, indem Sie zur CloudWatch-Konsole navigieren.

```
$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
```

Beachten Sie Folgendes:

- Die zum Initialisieren von `$dat.Timestamp` verwendeten Zeitstempeldaten müssen als UTC-Zeit (Coordinated Universal Time) angegeben werden.
- Der Wert, den Sie zum Initialisieren von `$dat.Value` verwenden, kann eine in Anführungszeichen angegebene Zeichenfolge oder ein numerischer Wert (ohne Anführungszeichen) sein. Das Beispiel zeigt einen Zeichenfolgenwert.

Weitere Informationen finden Sie auch unter:

- Arbeiten mit AWS-Services in AWS Tools for PowerShell
- https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/CloudWatch/MCloudWatchPutMetricDataPutMetricDataRequest.htmlAmazonCloudWatchClient.PutMetricData (.NET SDK-Referenz)
- https://docs.aws.amazon.com/AmazonCloudWatch/latest/APIReference/API_MetricDatum.htmlMetricDatum (Service-API-Referenz)
- Amazon CloudWatch-Konsole

Verwenden des ClientConfig-Parameters in Cmdlets

Mit dem `ClientConfig`-Parameter können bestimmte Konfigurationseinstellungen angegeben werden, wenn Sie eine Verbindung zu einem Service herstellen. Die meisten möglichen Eigenschaften dieses Parameters sind in der `Amazon.Runtime.ClientConfig`-Klasse definiert, die an die APIs für AWS-Services vererbt wird. Ein Beispiel für einfache Vererbung finden Sie in der `Amazon.Keyspaces.AmazonKeyspacesConfig`-Klasse. Darüber hinaus definieren einige Services zusätzliche Eigenschaften, die nur für diesen Service geeignet sind. Ein Beispiel für zusätzliche Eigenschaften, die definiert wurden, finden Sie in der `Amazon.S3.AmazonS3Config`-Klasse, insbesondere in der `ForcePathStyle`-Eigenschaft.

Verwenden des **`ClientConfig`**-Parameters

Wenn Sie den `ClientConfig`-Parameter verwenden möchten, können Sie ihn in der Befehlszeile als `ClientConfig`-Objekt angeben oder PowerShell-Splatting verwenden, um eine Sammlung von Parameterwerten als Einheit an einen Befehl zu übergeben. Diese Methoden werden im folgenden Beispiel verdeutlicht. In den Beispielen wird davon ausgegangen, dass das `AWS.Tools.S3`-Modul installiert und importiert wurde und dass Sie über ein `[default]`-Anmeldeinformationsprofil mit entsprechenden Berechtigungen verfügen.

Definieren eines **`ClientConfig`**-Objekts

```
$s3Config = New-Object -TypeName Amazon.S3.AmazonS3Config
$s3Config.ForcePathStyle = $true
$s3Config.Timeout = [TimeSpan]::FromMilliseconds(150000)
Get-S3Object -BucketName <BUCKET_NAME> -ClientConfig $s3Config
```

Hinzufügen von **`ClientConfig`**-Eigenschaften mithilfe von PowerShell-Splatting

```
$params=@{
    ClientConfig=@{
        ForcePathStyle=$true
        Timeout=[TimeSpan]::FromMilliseconds(150000)
    }
    BucketName="<BUCKET_NAME>"
}

Get-S3Object @params
```

Verwenden einer undefinierten Eigenschaft

Wenn Sie PowerShell-Splatting verwenden und eine `ClientConfig`-Eigenschaft angeben, die nicht existiert, erkennt AWS Tools for PowerShell den Fehler erst zur Laufzeit und gibt dann eine Ausnahme zurück. Ändern des obigen Beispiels:

```
$params=@{
    ClientConfig=@{
        ForcePathStyle=$true
        UndefinedProperty="Value"
        Timeout=[TimeSpan]::FromMilliseconds(150000)
    }
    BucketName="<BUCKET_NAME>"
}

Get-S3Object @params
```

In diesem Beispiel wird eine Ausnahme in etwa wie folgt erstellt:

```
Cannot bind parameter 'ClientConfig'. Cannot create object of type
 "Amazon.S3.AmazonS3Config". The UndefinedProperty property was not found for the
 Amazon.S3.AmazonS3Config object.
```

Angeben der AWS-Region

Mit dem `ClientConfig`-Parameter können Sie die AWS-Region für den Befehl festlegen. Die Region wird über die `RegionEndpoint`-Eigenschaft festgelegt. AWS Tools for PowerShell berechnet die zu verwendende Region gemäß der folgenden Rangfolge:

1. Der `-Region`-Parameter
2. Die Region, die im `ClientConfig`-Parameter übergeben wurde
3. Der Status der PowerShance-Sitzung
4. Die geteilte `config`-Datei von AWS
5. Die Umgebungsvariablen
6. Die Instance-Metadaten von Amazon-EC2, sofern aktiviert.

Tools für PowerShell Codebeispiele

Die Codebeispiele in diesem Thema zeigen Ihnen, wie Sie AWS Tools for PowerShell with verwenden AWS.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Serviceübergreifende Beispiele sind Beispielanwendungen, die über mehrere AWS-Services hinweg arbeiten.

Beispiele

- Aktionen und Szenarien mit Tools für PowerShell

Aktionen und Szenarien mit Tools für PowerShell

Die folgenden Codebeispiele zeigen, wie Aktionen ausgeführt und allgemeine Szenarien mithilfe von with implementiert werden AWS-Services. AWS Tools for PowerShell

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Services

- ACM-Beispiele mit Tools für PowerShell
- AppStream 2.0-Beispiele mit Tools für PowerShell
- Aurora-Beispiele für die Verwendung von Tools für PowerShell
- Auto Scaling Scaling-Beispiele mit Tools für PowerShell
- AWS Budgets Beispiele für die Verwendung von Tools für PowerShell

- AWS Cloud9 Beispiele für die Verwendung von Tools für PowerShell
- AWS CloudFormation Beispiele für die Verwendung von Tools für PowerShell
- CloudFront Beispiele für die Verwendung von Tools für PowerShell
- CloudTrail Beispiele für die Verwendung von Tools für PowerShell
- CloudWatch Beispiele für die Verwendung von Tools für PowerShell
- CodeCommit Beispiele für die Verwendung von Tools für PowerShell
- CodeDeploy Beispiele für die Verwendung von Tools für PowerShell
- CodePipeline Beispiele für die Verwendung von Tools für PowerShell
- Beispiele für Amazon Cognito Identity mit Tools für PowerShell
- AWS Config Beispiele für die Verwendung von Tools für PowerShell
- Beispiele für Device Farm mit Tools für PowerShell
- AWS Directory Service Beispiele für die Verwendung von Tools für PowerShell
- AWS DMS Beispiele für die Verwendung von Tools für PowerShell
- DynamoDB-Beispiele mit Tools für PowerShell
- Amazon EC2 EC2-Beispiele mit Tools für PowerShell
- Amazon ECR-Beispiele mit Tools für PowerShell
- Amazon ECS-Beispiele für die Verwendung von Tools für PowerShell
- Amazon EFS-Beispiele für die Verwendung von Tools für PowerShell
- Amazon EKS-Beispiele für die Verwendung von Tools für PowerShell
- Elastic Load Balancing — Beispiele für Version 1 mit Tools für PowerShell
- Elastic Load Balancing — Beispiele für Version 2 mit Tools für PowerShell
- Amazon FSx-Beispiele mit Tools für PowerShell
- AWS Glue Beispiele für die Verwendung von Tools für PowerShell
- AWS Health Beispiele für die Verwendung von Tools für PowerShell
- IAM-Beispiele mit Tools für PowerShell
- Kinesis-Beispiele für die Verwendung von Tools für PowerShell
- Lambda-Beispiele mit Tools für PowerShell
- Amazon ML-Beispiele mit Tools für PowerShell
- Macie-Beispiele mit Tools für PowerShell

- AWS OpsWorks Beispiele für die Verwendung von Tools für PowerShell
- AWS-Preisliste Beispiele für die Verwendung von Tools für PowerShell
- Beispiele für Resource Groups mit Tools für PowerShell
- API-Beispiele für das Tagging von Resource Groups mithilfe von Tools für PowerShell
- Route 53-Beispiele mit Tools für PowerShell
- Amazon S3 S3-Beispiele mit Tools für PowerShell
- S3 Glacier-Beispiele mit Tools für PowerShell
- Amazon SES SES-Beispiele für die Verwendung von Tools für PowerShell
- Amazon SNS SNS-Beispiele für die Verwendung von Tools für PowerShell
- Amazon SQS SQS-Beispiele mit Tools für PowerShell
- AWS STS Beispiele für die Verwendung von Tools für PowerShell
- AWS Support Beispiele für die Verwendung von Tools für PowerShell
- Systems Manager Manager-Beispiele mit Tools für PowerShell
- Amazon Translate Translate-Beispiele mit Tools für PowerShell
- AWS WAFV2 Beispiele für die Verwendung von Tools für PowerShell
- WorkSpaces Beispiele für die Verwendung von Tools für PowerShell

ACM-Beispiele mit Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS Tools for PowerShell mit ACM Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`Get-ACMCertificate`

Das folgende Codebeispiel zeigt, wie Sie es verwenden`Get-ACMCertificate`.

Tools für PowerShell

Beispiel 1: Dieses Beispiel zeigt, wie ein Zertifikat und seine Kette mithilfe des ARN des Zertifikats zurückgegeben werden.

```
Get-ACMCertificate -CertificateArn "arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
```

- Einzelheiten zur API finden Sie unter GetCertificate AWS Tools for PowerShellCmdlet-Referenz.

`Get-ACMCertificateDetail`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ACMCertificateDetail`

Tools für PowerShell

Beispiel 1: Gibt Details des angegebenen Zertifikats zurück.

```
Get-ACMCertificateDetail -CertificateArn "arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
```

Ausgabe:

```
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}
```

- Einzelheiten zur API finden Sie unter DescribeCertificate AWS Tools for PowerShellCmdlet-Referenz.

`Get-ACMCertificateList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ACMCertificateList`

Tools für PowerShell

Beispiel 1: Ruft eine Liste aller Ihrer Zertifikat-ARNs und deren Domainnamen ab. Das Cmdlet paginiert automatisch, um alle ARNs abzurufen. Um die Paginierung manuell zu steuern, verwenden Sie den MaxItem Parameter -, um zu steuern, wie viele Zertifikat-ARNs für jeden Serviceaufruf zurückgegeben werden, und den NextToken Parameter -, um den Startpunkt für jeden Aufruf anzugeben.

```
Get-ACMCertificateList
```

Ausgabe:

```
CertificateArn
 DomainName
--------------
 ----------
arn:aws:acm:us-east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
 www.example.com
```

Beispiel 2: Ruft eine Liste all Ihrer Zertifikat-ARNs ab, deren Zertifikatsstatus den angegebenen Status entspricht.

```
Get-ACMCertificateList -CertificateStatus "VALIDATION_TIMED_OUT","FAILED"
```

Beispiel 3: Dieses Beispiel gibt eine Liste aller Zertifikate in der Region us-east-1 zurück, die den Schlüsseltyp RSA_2048 und die erweiterte Schlüsselverwendung oder den Zweck von

CODE_SIGNING haben. Die Werte für diese Filterparameter finden Sie im Referenzthema Filters API: https://docs.aws.amazon.com/acm/latest/APIReference/API_Filters.html. ListCertificates

```
Get-ACMCertificateList -Region us-east-1 -Includes_KeyType RSA_2048 -
Includes_ExtendedKeyUsage CODE_SIGNING
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter ListCertificates AWS Tools for PowerShellCmdlet-Referenz.

`New-ACMCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `New-ACMCertificate`

Tools für PowerShell

Beispiel 1: Erstellt ein neues Zertifikat. Der Dienst gibt den ARN des neuen Zertifikats zurück.

```
New-ACMCertificate -DomainName "www.example.com"
```

Ausgabe:

```
arn:aws:acm:us-east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
```

Beispiel 2: Erstellt ein neues Zertifikat. Der Dienst gibt den ARN des neuen Zertifikats zurück.

```
New-ACMCertificate -DomainName "www.example.com" -SubjectAlternativeName
 "example.com","www.example.net"
```

Ausgabe:

```
arn:aws:acm:us-east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
```

- Einzelheiten zur API finden Sie unter RequestCertificate AWS Tools for PowerShellCmdlet-Referenz.

`Remove-ACMCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ACMCertificate`

Tools für PowerShell

Beispiel 1: Löscht das Zertifikat, das durch den angegebenen ARN und den zugehörigen privaten Schlüssel identifiziert wurde. Das Cmdlet fordert Sie zur Bestätigung auf, bevor Sie fortfahren. Fügen Sie die Option -Force hinzu, um die Bestätigung zu unterdrücken.

```
Remove-ACMCertificate -CertificateArn "arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
```

- Einzelheiten zur API finden Sie unter DeleteCertificateCmdlet-Referenz.AWS Tools for PowerShell

`Send-ACMValidationEmail`

Das folgende Codebeispiel zeigt die Verwendung. `Send-ACMValidationEmail`

Tools für PowerShell

Beispiel 1: Fordert an, dass die E-Mail zur Bestätigung des Domainbesitzes für „www.example.com“ gesendet wird. Wenn $ in Ihrer Shell auf „Mittel“ oder niedriger gesetzt

ConfirmPreference ist, fordert das Cmdlet vor dem Fortfahren zur Bestätigung auf. Fügen Sie die Option -Force hinzu, um Bestätigungsaufforderungen zu unterdrücken.

```
$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
```

- Einzelheiten zur API finden Sie unter ResendValidationEmail AWS Tools for PowerShellCmdlet-Referenz.

AppStream 2.0-Beispiele mit Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS Tools for PowerShell mit AppStream 2.0 Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`Add-APSResourceTag`

Das folgende Codebeispiel zeigt, wie Sie es verwendenAdd-APSResourceTag.

Tools für PowerShell

Beispiel 1: Dieses Beispiel fügt der Ressource ein Ressourcen-Tag hinzu AppStream

```
Add-APSResourceTag -ResourceArn arn:aws:appstream:us-east-1:123456789012:stack/
SessionScriptTest -Tag @{StackState='Test'} -Select ^Tag
```

Ausgabe:

```
Name                           Value
----                           -----
StackState                     Test
```

- Einzelheiten zur API finden Sie unter TagResource AWS Tools for PowerShellCmdlet-Referenz.

`Copy-APSImage`

Das folgende Codebeispiel zeigt die Verwendung. `Copy-APSImage`

Tools für PowerShell

Beispiel 1: Dieses Beispiel kopiert ein Bild in eine andere Region

```
Copy-APSImage -DestinationImageName TestImageCopy -DestinationRegion us-west-2 -
SourceImageName Powershell
```

Ausgabe:

```
TestImageCopy
```

- Einzelheiten zur API finden Sie unter CopyImage AWS Tools for PowerShellCmdlet-Referenz.

`Disable-APSUSer`

Das folgende Codebeispiel zeigt die Verwendung. `Disable-APSUSer`

Tools für PowerShell

Beispiel 1: Dieses Beispiel deaktiviert einen Benutzer in USERPOOL

```
Disable-APSUser -AuthenticationType USERPOOL -UserName TestUser@lab.com
```

- Einzelheiten zur API finden Sie unter DisableUser AWS Tools for PowerShellCmdlet-Referenz.

`Enable-APSUser`

Das folgende Codebeispiel zeigt die Verwendung. `Enable-APSUser`

Tools für PowerShell

Beispiel 1: Dieses Beispiel aktiviert einen deaktivierten Benutzer in USERPOOL

```
Enable-APSUser -AuthenticationType USERPOOL -UserName TestUser@lab.com
```

- Einzelheiten zur API finden Sie unter EnableUser AWS Tools for PowerShellCmdlet-Referenz.

`Get-APSAssociatedFleetList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-APSAssociatedFleetList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Flotte angezeigt, die einem Stapel zugeordnet ist

```
Get-APSAssociatedFleetList -StackName PowershellStack
```

Ausgabe:

```
PowershellFleet
```

- Einzelheiten zur API finden Sie unter ListAssociatedFleets AWS Tools for PowerShellCmdlet-Referenz.

`Get-APSAssociatedStackList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-APSAssociatedStackList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Stapel angezeigt, der einer Flotte zugeordnet ist

```
Get-APSAssociatedStackList -FleetName PowershellFleet
```

Ausgabe:

```
PowershellStack
```

- Einzelheiten zur API finden Sie unter ListAssociatedStacks AWS Tools for PowerShellCmdlet-Referenz.

`Get-APSDirectoryConfigList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-APSDirectoryConfigList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Verzeichniskonfigurationen angezeigt, die in erstellt wurden AppStream

```
Get-APSDirectoryConfigList | Select DirectoryName,
 OrganizationalUnitDistinguishedNames, CreatedTime
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeDirectoryConfigs AWS Tools for PowerShellCmdlet-Referenz.

`Get-APSFleetList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-APSFleetList`

Tools für PowerShell

Beispiel 1: Dieses Beispiel zeigt Details einer Flotte

```
Get-APSFleetList -Name Test
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeFleets AWS Tools for PowerShellCmdlet-Referenz.

`Get-APSImageBuilderList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-APSImageBuilderList`

Tools für PowerShell

Beispiel 1: Dieses Beispiel zeigt Details eines ImageBuilder

```
Get-APSImageBuilderList -Name TestImage
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeImageBuilders AWS Tools for PowerShellCmdlet-Referenz.

`Get-APSImageList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-APSImageList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden private AppStream Bilder angezeigt

```
Get-APSImageList -Type PRIVATE | select DisplayName, ImageBuilderName, Visibility,
 arn
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeImages AWS Tools for PowerShellCmdlet-Referenz.

`Get-APSImagePermission`

Das folgende Codebeispiel zeigt die Verwendung. `Get-APSImagePermission`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Bildberechtigungen für ein geteiltes AppStream Bild angezeigt

```
Get-APSImagePermission -Name Powershell  | select SharedAccountId,
 @{n="AllowFleet";e={$_.ImagePermissions.AllowFleet}},
 @{n="AllowImageBuilder";e={$_.ImagePermissions.AllowImageBuilder}}
```

Ausgabe:

```
SharedAccountId AllowFleet AllowImageBuilder
--------------- ---------- -----------------
123456789012          True              True
```

- Einzelheiten zur API finden Sie unter DescribeImagePermissions AWS Tools for PowerShellCmdlet-Referenz.

`Get-APSSessionList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-APSSessionList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste von Sitzungen für eine Flotte angezeigt

```
Get-APSSessionList -FleetName PowershellFleet -StackName PowershellStack
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeSessions AWS Tools for PowerShellCmdlet-Referenz.

`Get-APSStackList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-APSStackList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste von AppStream Stack angezeigt

```
Get-APSStackList | Select DisplayName, Arn, CreatedTime
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeStacks AWS Tools for PowerShellCmdlet-Referenz.

`Get-APSTagsForResourceList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-APSTagsForResourceList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Tags auf einer AppStream Ressource angezeigt

```
Get-APSTagsForResourceList -ResourceArn arn:aws:appstream:us-
east-1:123456789012:stack/SessionScriptTest
```

Ausgabe:

```
Key        Value
---        -----
StackState Test
```

- Einzelheiten zur API finden Sie unter ListTagsForResource AWS Tools for PowerShellCmdlet-Referenz.

`Get-APSUsageReportSubscription`

Das folgende Codebeispiel zeigt die Verwendung. `Get-APSUsageReportSubscription`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden AppStreamUsageReport Konfigurationsdetails angezeigt

```
Get-APSUsageReportSubscription
```

Ausgabe:

```
LastGeneratedReportDate S3BucketName                                    Schedule
 SubscriptionErrors
----------------------- ------------                                    --------
 ------------------
1/1/0001 12:00:00 AM    appstream-logs-us-east-1-123456789012-sik1hnxe DAILY    {}
```

- Einzelheiten zur API finden Sie unter DescribeUsageReportSubscriptions AWS Tools for PowerShellCmdlet-Referenz.

`Get-APSUser`

Das folgende Codebeispiel zeigt die Verwendung. `Get-APSUser`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste von Benutzern mit aktiviertem Status angezeigt

```
Get-APSUser -AuthenticationType USERPOOL | Select-Object UserName,
 AuthenticationType, Enabled
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeUsers AWS Tools for PowerShellCmdlet-Referenz.

`Get-APSUserStackAssociation`

Das folgende Codebeispiel zeigt die Verwendung. `Get-APSUserStackAssociation`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste der Benutzer angezeigt, die einem Stack zugewiesen sind

```
Get-APSUserStackAssociation -StackName PowershellStack
```

Ausgabe:

```
AuthenticationType SendEmailNotification StackName       UserName
------------------ --------------------- ---------       --------
USERPOOL           False                 PowershellStack TestUser1@lab.com
USERPOOL           False                 PowershellStack TestUser2@lab.com
```

- Einzelheiten zur API finden Sie unter DescribeUserStackAssociations AWS Tools for PowerShellCmdlet-Referenz.

`New-APSDirectoryConfig`

Das folgende Codebeispiel zeigt die Verwendung. `New-APSDirectoryConfig`

Tools für PowerShell

Beispiel 1: Dieses Beispiel erstellt eine Verzeichniskonfiguration in AppStream

```
New-APSDirectoryConfig -ServiceAccountCredentials_AccountName contoso\ServiceAccount
 -ServiceAccountCredentials_AccountPassword MyPass -DirectoryName contoso.com -
OrganizationalUnitDistinguishedName "OU=AppStream,OU=Contoso,DC=Contoso,DC=com"
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter CreateDirectoryConfig AWS Tools for PowerShellCmdlet-Referenz.

`New-APSFleet`

Das folgende Codebeispiel zeigt die Verwendung. `New-APSFleet`

Tools für PowerShell

Beispiel 1: Dieses Beispiel erstellt eine neue 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
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter CreateFleet AWS Tools for PowerShellCmdlet-Referenz.

`New-APSImageBuilder`

Das folgende Codebeispiel zeigt die Verwendung. `New-APSImageBuilder`

Tools für PowerShell

Beispiel 1: Dieses Beispiel erstellt einen Image Builder in 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
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter CreateImageBuilder AWS Tools for PowerShellCmdlet-Referenz.

`New-APSImageBuilderStreamingURL`

Das folgende Codebeispiel zeigt die Verwendung. `New-APSImageBuilderStreamingURL`

Tools für PowerShell

Beispiel 1: Dieses Beispiel erstellt eine ImageBuilder Streaming-URL mit einer Gültigkeit von 2 Stunden

```
New-APSImageBuilderStreamingURL -Name TestIB -Validity 7200 -Region us-west-2
```

Ausgabe:

```
Expires              StreamingURL
-------              ------------
12/27/2019 1:49:13 PM https://appstream2.us-west-2.aws.amazon.com/authenticate?
parameters=eyJ0eXBlIjoiQURNSU4iLCJleHBpcmVzIjoiMTU3NzQ1NDU1MyIsImF3c0FjY291bnRJZCI6IjM5MzQwM
```

- Einzelheiten zur API finden Sie unter CreateImageBuilderStreamingURL in der AWS Tools for PowerShell Cmdlet-Referenz.

`New-APSStack`

Das folgende Codebeispiel zeigt die Verwendung. `New-APSStack`

Tools für PowerShell

Beispiel 1: Dieses Beispiel erstellt einen neuen AppStream Stack

```
New-APSStack -Name TestStack -DisplayName TestStack -ApplicationSettings_Enabled
 $True -ApplicationSettings_SettingsGroup TestStack -Region us-west-2
```

Ausgabe:

```
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}
```

- Einzelheiten zur API finden Sie unter CreateStack AWS Tools for PowerShellCmdlet-Referenz.

`New-APSStreamingURL`

Das folgende Codebeispiel zeigt die Verwendung. `New-APSStreamingURL`

Tools für PowerShell

Beispiel 1: Dieses Beispiel erstellt eine Streaming-URL von Stack

```
New-APSStreamingURL -StackName SessionScriptTest -FleetName SessionScriptNew -UserId
 TestUser
```

Ausgabe:

```
Expires                StreamingURL
```

```
-------                  ------------
12/27/2019 12:43:37 PM https://appstream2.us-east-1.aws.amazon.com/authenticate?
parameters=eyJ0eXBlIjoiRU5EX1VTRVIiLCJleHBpcmVzIjoiMTU3NzQ1MDYxNyIsImF3c0FjY291bnRJZCI6IjM5M
```

- Einzelheiten zur API finden Sie unter CreateStreamingURL in der AWS Tools for PowerShell Cmdlet-Referenz.

`New-APSUsageReportSubscription`

Das folgende Codebeispiel zeigt die Verwendung. `New-APSUsageReportSubscription`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden AppStream Nutzungsberichte aktiviert

```
New-APSUsageReportSubscription
```

Ausgabe:

```
S3BucketName                                    Schedule
------------                                    --------
appstream-logs-us-east-1-123456789012-sik2hnxe DAILY
```

- Einzelheiten zur API finden Sie unter CreateUsageReportSubscription AWS Tools for PowerShellCmdlet-Referenz.

`New-APSUser`

Das folgende Codebeispiel zeigt die Verwendung. `New-APSUser`

Tools für PowerShell

Beispiel 1: Dieses Beispiel erstellt einen Benutzer in USERPOOL

```
New-APSUser -UserName Test@lab.com -AuthenticationType USERPOOL -FirstName 'kt' -
LastName 'aws' -Select ^UserName
```

Ausgabe:

```
Test@lab.com
```

- Einzelheiten zur API finden Sie unter CreateUser AWS Tools for PowerShellCmdlet-Referenz.

`Register-APSFleet`

Das folgende Codebeispiel zeigt die Verwendung. `Register-APSFleet`

Tools für PowerShell

Beispiel 1: Dieses Beispiel registriert eine Flotte mit einem Stack

```
Register-APSFleet -StackName TestStack -FleetName TestFleet -Region us-west-2
```

- Einzelheiten zur API finden Sie unter AssociateFleet AWS Tools for PowerShellCmdlet-Referenz.

`Register-APSUserStackBatch`

Das folgende Codebeispiel zeigt die Verwendung. `Register-APSUserStackBatch`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird einem Benutzer in USERPOOL ein Stack zugewiesen

```
Register-APSUserStackBatch -UserStackAssociation
 @{AuthenticationType="USERPOOL";SendEmailNotification=
$False;StackName="PowershellStack";UserName="TestUser1@lab.com"}
```

- Einzelheiten zur API finden Sie unter BatchAssociateUserStack AWS Tools for PowerShellCmdlet-Referenz.

`Remove-APSDirectoryConfig`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-APSDirectoryConfig`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die AppStream Verzeichniskonfiguration entfernt

```
Remove-APSDirectoryConfig -DirectoryName contoso.com
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DeleteDirectoryConfig AWS Tools for PowerShellCmdlet-Referenz.

`Remove-APSFleet`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-APSFleet`

Tools für PowerShell

Beispiel 1: Dieses Beispiel entfernt und löscht eine Flotte AppStream

```
Remove-APSFleet -Name TestFleet -Region us-west-2
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DeleteFleet AWS Tools for PowerShellCmdlet-Referenz.

`Remove-APSImage`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-APSImage`

Tools für PowerShell

Beispiel 1: Dieses Beispiel löscht ein Bild

```
Remove-APSImage -Name TestImage -Region us-west-2
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DeleteImage AWS Tools for PowerShellCmdlet-Referenz.

`Remove-APSImageBuilder`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-APSImageBuilder`

Tools für PowerShell

Beispiel 1: Dieses Beispiel löscht ein ImageBuilder

```
Remove-APSImageBuilder -Name TestIB -Region us-west-2
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DeleteImageBuilder AWS Tools for PowerShellCmdlet-Referenz.

`Remove-APSImagePermission`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-APSImagePermission`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Berechtigungen eines Bilds entfernt

```
Remove-APSImagePermission -Name Powershell -SharedAccountId 123456789012
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DeleteImagePermissions AWS Tools for PowerShellCmdlet-Referenz.

`Remove-APSResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-APSResourceTag`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Ressourcen-Tag aus einer AppStream Ressource entfernt

```
Remove-APSResourceTag -ResourceArn arn:aws:appstream:us-east-1:123456789012:stack/
SessionScriptTest -TagKey StackState
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter UntagResource AWS Tools for PowerShellCmdlet-Referenz.

`Remove-APSStack`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-APSStack`

Tools für PowerShell

Beispiel 1: Dieses Beispiel löscht einen Stack

```
Remove-APSStack -Name TestStack -Region us-west-2
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DeleteStack AWS Tools for PowerShellCmdlet-Referenz.

`Remove-APSUsageReportSubscription`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-APSUsageReportSubscription`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Abonnement für AppStream Nutzungsberichte deaktiviert

```
Remove-APSUsageReportSubscription
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DeleteUsageReportSubscription AWS Tools for PowerShellCmdlet-Referenz.

`Remove-APSUser`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-APSUser`

Tools für PowerShell

Beispiel 1: Dieses Beispiel löscht einen Benutzer aus USERPOOL

```
Remove-APSUser -UserName TestUser@lab.com -AuthenticationType USERPOOL
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DeleteUser AWS Tools for PowerShellCmdlet-Referenz.

`Revoke-APSSession`

Das folgende Codebeispiel zeigt die Verwendung. `Revoke-APSSession`

Tools für PowerShell

Beispiel 1: Dieses Beispiel widerruft eine Sitzung für Fleet AppStream

```
Revoke-APSSession -SessionId 6cd2f9a3-f948-4aa1-8014-8a7dcde14877
```

- Einzelheiten zur API finden Sie unter ExpireSession AWS Tools for PowerShellCmdlet-Referenz.

`Start-APSFleet`

Das folgende Codebeispiel zeigt die Verwendung. `Start-APSFleet`

Tools für PowerShell

Beispiel 1: Dieses Beispiel startet eine Flotte

```
Start-APSFleet -Name PowershellFleet
```

- Einzelheiten zur API finden Sie unter StartFleet AWS Tools for PowerShellCmdlet-Referenz.

`Start-APSImageBuilder`

Das folgende Codebeispiel zeigt die Verwendung. `Start-APSImageBuilder`

Tools für PowerShell

Beispiel 1: Dieses Beispiel startet ein ImageBuilder

```
Start-APSImageBuilder -Name TestImage
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter StartImageBuilder AWS Tools for PowerShellCmdlet-Referenz.

`Stop-APSFleet`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-APSFleet`

Tools für PowerShell

Beispiel 1: Dieses Beispiel stoppt eine Flotte

```
Stop-APSFleet -Name PowershellFleet
```

- Einzelheiten zur API finden Sie unter StopFleet AWS Tools for PowerShellCmdlet-Referenz.

`Stop-APSImageBuilder`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-APSImageBuilder`

Tools für PowerShell

Beispiel 1: Dieses Beispiel stoppt ein ImageBuilder

```
Stop-APSImageBuilder -Name TestImage
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter StopImageBuilder AWS Tools for PowerShellCmdlet-Referenz.

`Unregister-APSFleet`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-APSFleet`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Registrierung einer Flotte vom Stapel aufgehoben

```
Unregister-APSFleet -StackName TestStack -FleetName TestFleet -Region us-west-2
```

- Einzelheiten zur API finden Sie unter DisassociateFleet AWS Tools for PowerShellCmdlet-Referenz.

`Unregister-APSUserStackBatch`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-APSUserStackBatch`

Tools für PowerShell

Beispiel 1: Dieses Beispiel entfernt einen Benutzer aus einem zugewiesenen Stack

```
Unregister-APSUserStackBatch -UserStackAssociation
 @{AuthenticationType="USERPOOL";SendEmailNotification=
$False;StackName="PowershellStack";UserName="TestUser1@lab.com"}
```

- Einzelheiten zur API finden Sie unter BatchDisassociateUserStack AWS Tools for PowerShellCmdlet-Referenz.

`Update-APSDirectoryConfig`

Das folgende Codebeispiel zeigt die Verwendung. `Update-APSDirectoryConfig`

Tools für PowerShell

Beispiel 1: Dieses Beispiel aktualisiert die Verzeichniskonfiguration, die in erstellt wurde AppStream

```
Update-APSDirectoryConfig -ServiceAccountCredentials_AccountName contoso
\ServiceAccount -ServiceAccountCredentials_AccountPassword MyPass@1$@#
 -DirectoryName contoso.com -OrganizationalUnitDistinguishedName
 "OU=AppStreamNew,OU=Contoso,DC=Contoso,DC=com"
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter UpdateDirectoryConfig AWS Tools for PowerShellCmdlet-Referenz.

`Update-APSFleet`

Das folgende Codebeispiel zeigt die Verwendung. `Update-APSFleet`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Eigenschaften einer Flotte aktualisiert

```
Update-APSFleet -Name PowershellFleet -EnableDefaultInternetAccess $True -
DisconnectTimeoutInSecond 950
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter UpdateFleet AWS Tools for PowerShellCmdlet-Referenz.

`Update-APSImagePermission`

Das folgende Codebeispiel zeigt die Verwendung. `Update-APSImagePermission`

Tools für PowerShell

Beispiel 1: Dieses Beispiel teilt ein AppStream Bild mit einem anderen Konto

```
Update-APSImagePermission -Name Powershell -SharedAccountId 123456789012 -
ImagePermissions_AllowFleet $True -ImagePermissions_AllowImageBuilder $True
```

- Einzelheiten zur API finden Sie unter UpdateImagePermissions AWS Tools for PowerShellCmdlet-Referenz.

`Update-APSStack`

Das folgende Codebeispiel zeigt die Verwendung. `Update-APSStack`

Tools für PowerShell

Beispiel 1: Dieses Beispiel aktualisiert (aktiviert) die Persistenz von Anwendungseinstellungen und Basisordnern auf einem Stack

```
Update-APSStack -Name PowershellStack -ApplicationSettings_Enabled $True
 -ApplicationSettings_SettingsGroup PowershellStack -StorageConnector
 @{ConnectorType="HOMEFOLDERS"}
```

Ausgabe:

```
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}
```

- Einzelheiten zur API finden Sie unter UpdateStack AWS Tools for PowerShellCmdlet-Referenz.

Aurora-Beispiele für die Verwendung von Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS Tools for PowerShell mit Aurora Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`Get-RDSOrderableDBInstanceOption`

Das folgende Codebeispiel zeigt, wie Sie es verwenden`Get-RDSOrderableDBInstanceOption`.

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die DB-Engine-Versionen aufgeführt, die eine bestimmte DB-Instance-Klasse in einem unterstützen AWS-Region.

```
$params = @{
  Engine = 'aurora-postgresql'
  DBInstanceClass = 'db.r5.large'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

Beispiel 2: In diesem Beispiel werden die DB-Instance-Klassen aufgeführt, die für eine bestimmte DB-Engine-Version in einem unterstützt werden AWS-Region.

```
$params = @{
  Engine = 'aurora-postgresql'
  EngineVersion = '13.6'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

- Einzelheiten zur API finden Sie unter DescribeOrderableDB InstanceOptions in AWS Tools for PowerShell Cmdlet Reference.

Auto Scaling Scaling-Beispiele mit Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS Tools for PowerShell mit Auto Scaling Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`Add-ASLoadBalancer`

Das folgende Codebeispiel zeigt, wie Sie es verwendenAdd-ASLoadBalancer.

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Load Balancer der angegebenen Auto Scaling Scaling-Gruppe zugeordnet.

```
Add-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter AttachLoadBalancers AWS Tools for PowerShellCmdlet-Referenz.

`Complete-ASLifecycleAction`

Das folgende Codebeispiel zeigt die Verwendung. Complete-ASLifecycleAction

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Lebenszyklusaktion abgeschlossen.

```
Complete-ASLifecycleAction -LifecycleHookName myLifecycleHook -
AutoScalingGroupName my-asg -LifecycleActionResult CONTINUE -LifecycleActionToken
 bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- Einzelheiten zur API finden Sie unter CompleteLifecycleAction AWS Tools for PowerShellCmdlet-Referenz.

`Disable-ASMetricsCollection`

Das folgende Codebeispiel zeigt die Verwendung. Disable-ASMetricsCollection

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Überwachung der angegebenen Metriken für die angegebene Auto Scaling Scaling-Gruppe deaktiviert.

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg -Metric @("GroupMinSize",
 "GroupMaxSize")
```

Beispiel 2: In diesem Beispiel wird die Überwachung aller Metriken für die angegebene Auto Scaling Scaling-Gruppe deaktiviert.

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter DisableMetricsCollection AWS Tools for PowerShellCmdlet-Referenz.

`Dismount-ASInstance`

Das folgende Codebeispiel zeigt die Verwendung. `Dismount-ASInstance`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instance von der angegebenen Auto Scaling-Gruppe getrennt und die gewünschte Kapazität verringert, sodass Auto Scaling keine Ersatz-Instance startet.

```
Dismount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

Ausgabe:

```
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        :
```

Beispiel 2: In diesem Beispiel wird die angegebene Instance von der angegebenen Auto Scaling Scaling-Gruppe getrennt, ohne die gewünschte Kapazität zu verringern. Auto Scaling startet eine Ersatzinstanz.

```
Dismount-ASInstance -InstanceId i-7bf746a2 -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

Ausgabe:

```
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        :
```

- Einzelheiten zur API finden Sie unter DetachInstances AWS Tools for PowerShellCmdlet-Referenz.

`Dismount-ASLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung. `Dismount-ASLoadBalancer`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Load Balancer von der angegebenen Auto Scaling Scaling-Gruppe getrennt.

```
Dismount-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter DetachLoadBalancers AWS Tools for PowerShellCmdlet-Referenz.

`Enable-ASMetricsCollection`

Das folgende Codebeispiel zeigt die Verwendung. `Enable-ASMetricsCollection`

Tools für PowerShell

Beispiel 1: Dieses Beispiel ermöglicht die Überwachung der angegebenen Metriken für die angegebene Auto Scaling Scaling-Gruppe.

```
Enable-ASMetricsCollection  -Metric @("GroupMinSize", "GroupMaxSize") -
AutoScalingGroupName my-asg -Granularity 1Minute
```

Beispiel 2: Dieses Beispiel ermöglicht die Überwachung aller Metriken für die angegebene Auto Scaling Scaling-Gruppe.

```
Enable-ASMetricsCollection -AutoScalingGroupName my-asg -Granularity 1Minute
```

- Einzelheiten zur API finden Sie unter EnableMetricsCollection AWS Tools for PowerShellCmdlet-Referenz.

`Enter-ASStandby`

Das folgende Codebeispiel zeigt die Verwendung. `Enter-ASStandby`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instance in den Standby-Modus versetzt und die gewünschte Kapazität verringert, sodass Auto Scaling keine Ersatz-Instance startet.

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

Ausgabe:

```
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        :
```

Beispiel 2: In diesem Beispiel wird die angegebene Instance in den Standby-Modus versetzt, ohne die gewünschte Kapazität zu verringern. Auto Scaling startet eine Ersatzinstanz.

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

Ausgabe:

```
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        :
```

- Einzelheiten zur API finden Sie unter EnterStandby AWS Tools for PowerShellCmdlet-Referenz.

`Exit-ASStandby`

Das folgende Codebeispiel zeigt die Verwendung. `Exit-ASStandby`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instanz aus dem Standby-Modus versetzt.

```
Exit-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

Ausgabe:

```
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        :
```

- Einzelheiten zur API finden Sie unter ExitStandby AWS Tools for PowerShellCmdlet-Referenz.

`Get-ASAccountLimit`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASAccountLimit`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Auto Scaling Scaling-Ressourcenlimits für Ihr AWS Konto beschrieben.

```
Get-ASAccountLimit
```

Ausgabe:

```
MaxNumberOfAutoScalingGroups    : 20
MaxNumberOfLaunchConfigurations : 100
```

- Einzelheiten zur API finden Sie unter DescribeAccountLimits AWS Tools for PowerShellCmdlet-Referenz.

`Get-ASAdjustmentType`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASAdjustmentType`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die Anpassungstypen, die von Auto Scaling unterstützt werden.

```
Get-ASAdjustmentType
```

Ausgabe:

```
Type
----
ChangeInCapacity
ExactCapacity
PercentChangeInCapacity
```

- Einzelheiten zur API finden Sie unter DescribeAdjustmentTypes AWS Tools for PowerShellCmdlet-Referenz.

`Get-ASAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASAutoScalingGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Namen Ihrer Auto Scaling Scaling-Gruppen aufgeführt.

```
Get-ASAutoScalingGroup | format-table -property AutoScalingGroupName
```

Ausgabe:

```
AutoScalingGroupName
--------------------
my-asg-1
my-asg-2
my-asg-3
my-asg-4
my-asg-5
my-asg-6
```

Beispiel 2: Dieses Beispiel beschreibt die angegebene Auto Scaling Scaling-Gruppe.

```
Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1
```

Ausgabe:

```
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
```

Beispiel 3: Dieses Beispiel beschreibt die angegebenen zwei Auto Scaling Scaling-Gruppen.

```
Get-ASAutoScalingGroup -AutoScalingGroupName @("my-asg-1", "my-asg-2")
```

Beispiel 4: Dieses Beispiel beschreibt die Auto Scaling Scaling-Instances für die angegebene Auto Scaling Scaling-Gruppe.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1).Instances
```

Beispiel 5: Dieses Beispiel beschreibt alle Ihre Auto Scaling Scaling-Gruppen.

```
Get-ASAutoScalingGroup
```

Beispiel 6: In diesem Beispiel werden alle Ihre Auto Scaling Scaling-Gruppen in Batches von 10 beschrieben.

```
$nextToken = $null
do {
  Get-ASAutoScalingGroup -NextToken $nextToken -MaxRecord 10
  $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

Beispiel 7: Dieses LaunchTemplate Beispiel beschreibt die angegebene Auto Scaling Scaling-Gruppe. In diesem Beispiel wird davon ausgegangen, dass die Option „Instance-Kaufoptionen“ auf „An der Startvorlage festhalten“ gesetzt ist. Falls diese Option auf „Kaufoptionen und Instanztypen kombinieren“ gesetzt ist, LaunchTemplate könnte mit "darauf zugegriffen werdenMixedInstancesPolicy. LaunchTemplate„Eigenschaft.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-ag-1).LaunchTemplate
```

Ausgabe:

```
LaunchTemplateId     LaunchTemplateName   Version
----------------     ------------------   -------
lt-06095fd619cb40371 test-launch-template $Default
```

- Einzelheiten zur API finden Sie unter DescribeAutoScalingGroups AWS Tools for PowerShellCmdlet-Referenz.

`Get-ASAutoScalingInstance`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASAutoScalingInstance`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die IDs Ihrer Auto Scaling Scaling-Instances aufgeführt.

```
Get-ASAutoScalingInstance | format-table -property InstanceId
```

Ausgabe:

```
InstanceId
----------
```

```
i-12345678
i-87654321
i-abcd1234
```

Beispiel 2: Dieses Beispiel beschreibt die angegebene Auto Scaling Scaling-Instanz.

```
Get-ASAutoScalingInstance -InstanceId i-12345678
```

Ausgabe:

```
AutoScalingGroupName    : my-asg
AvailabilityZone        : us-west-2b
HealthStatus            : HEALTHY
InstanceId              : i-12345678
LaunchConfigurationName : my-lc
LifecycleState          : InService
```

Beispiel 3: Dieses Beispiel beschreibt die angegebenen zwei Auto Scaling Scaling-Instances.

```
Get-ASAutoScalingInstance -InstanceId @("i-12345678", "i-87654321")
```

Beispiel 4: Dieses Beispiel beschreibt die Auto Scaling Scaling-Instances für die angegebene Auto Scaling Scaling-Gruppe.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg).Instances | Get-
ASAutoScalingInstance
```

Beispiel 5: Dieses Beispiel beschreibt alle Ihre Auto Scaling Scaling-Instances.

```
Get-ASAutoScalingInstance
```

Beispiel 6: In diesem Beispiel werden alle Ihre Auto Scaling Scaling-Instances in Batches von 10 beschrieben.

```
$nextToken = $null
do {
  Get-ASAutoScalingInstance -NextToken $nextToken -MaxRecord 10
  $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Einzelheiten zur API finden Sie unter DescribeAutoScalingInstances AWS Tools for PowerShellCmdlet-Referenz.

`Get-ASAutoScalingNotificationType`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASAutoScalingNotificationType`

Tools für PowerShell

Beispiel 1: Dieses Beispiel listet die Benachrichtigungstypen auf, die von Auto Scaling unterstützt werden.

```
Get-ASAutoScalingNotificationType
```

Ausgabe:

```
autoscaling:EC2_INSTANCE_LAUNCH
autoscaling:EC2_INSTANCE_LAUNCH_ERROR
autoscaling:EC2_INSTANCE_TERMINATE
autoscaling:EC2_INSTANCE_TERMINATE_ERROR
autoscaling:TEST_NOTIFICATION
```

- Einzelheiten zur API finden Sie unter DescribeAutoScalingNotificationTypes AWS Tools for PowerShellCmdlet-Referenz.

`Get-ASLaunchConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASLaunchConfiguration`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Namen Ihrer Startkonfigurationen aufgeführt.

```
Get-ASLaunchConfiguration | format-table -property LaunchConfigurationName
```

Ausgabe:

```
LaunchConfigurationName
-----------------------
```

```
my-lc-1
my-lc-2
my-lc-3
my-lc-4
my-lc-5
```

Beispiel 2: Dieses Beispiel beschreibt die angegebene Startkonfiguration.

```
Get-ASLaunchConfiguration -LaunchConfigurationName my-lc-1
```

Ausgabe:

```
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                     :
```

Beispiel 3: Dieses Beispiel beschreibt die beiden angegebenen Startkonfigurationen.

```
Get-ASLaunchConfiguration -LaunchConfigurationName @("my-lc-1", "my-lc-2")
```

Beispiel 4: Dieses Beispiel beschreibt all Ihre Startkonfigurationen.

```
Get-ASLaunchConfiguration
```

Beispiel 5: Dieses Beispiel beschreibt all Ihre Startkonfigurationen in Batches von 10 Stück.

```
$nextToken = $null
do {
  Get-ASLaunchConfiguration -NextToken $nextToken -MaxRecord 10
  $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Einzelheiten zur API finden Sie unter DescribeLaunchConfigurations AWS Tools for PowerShellCmdlet-Referenz.

`Get-ASLifecycleHook`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASLifecycleHook`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt den angegebenen Lifecycle-Hook.

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName myLifecycleHook
```

Ausgabe:

```
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
```

Beispiel 2: Dieses Beispiel beschreibt alle Lifecycle-Hooks für die angegebene Auto Scaling Scaling-Gruppe.

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg
```

Beispiel 3: In diesem Beispiel werden alle Lifecycle-Hooks für all Ihre Auto Scaling Scaling-Gruppen beschrieben.

```
Get-ASLifecycleHook
```

- Einzelheiten zur API finden Sie unter DescribeLifecycleHooks AWS Tools for PowerShellCmdlet-Referenz.

`Get-ASLifecycleHookType`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASLifecycleHookType`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die von Auto Scaling unterstützten Lifecycle-Hook-Typen aufgeführt.

```
Get-ASLifecycleHookType
```

Ausgabe:

```
autoscaling:EC2_INSTANCE_LAUNCHING
auto-scaling:EC2_INSTANCE_TERMINATING
```

- Einzelheiten zur API finden Sie unter DescribeLifecycleHookTypes AWS Tools for PowerShellCmdlet-Referenz.

`Get-ASLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASLoadBalancer`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die Load Balancer für die angegebene Auto Scaling Scaling-Gruppe.

```
Get-ASLoadBalancer -AutoScalingGroupName my-asg
```

Ausgabe:

```
LoadBalancerName    State
----------------    -----
```

```
my-lb                Added
```

- Einzelheiten zur API finden Sie unter DescribeLoadBalancers AWS Tools for PowerShellCmdlet-Referenz.

`Get-ASMetricCollectionType`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASMetricCollectionType`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Arten der Metrikerfassung aufgeführt, die von Auto Scaling unterstützt werden.

```
(Get-ASMetricCollectionType).Metrics
```

Ausgabe:

```
Metric
------
GroupMinSize
GroupMaxSize
GroupDesiredCapacity
GroupInServiceInstances
GroupPendingInstances
GroupTerminatingInstances
GroupStandbyInstances
GroupTotalInstances
```

Beispiel 2: In diesem Beispiel werden die entsprechenden Granularitäten aufgeführt.

```
(Get-ASMetricCollectionType).Granularities
```

Ausgabe:

```
Granularity
-----------
1Minute
```

- Einzelheiten zur API finden Sie unter DescribeMetricCollectionTypes AWS Tools for PowerShellCmdlet-Referenz.

`Get-ASNotificationConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASNotificationConfiguration`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die Benachrichtigungsaktionen, die mit der angegebenen Auto Scaling Scaling-Gruppe verknüpft sind.

```
Get-ASNotificationConfiguration -AutoScalingGroupName my-asg | format-list
```

Ausgabe:

```
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
```

Beispiel 2: In diesem Beispiel werden die Benachrichtigungsaktionen beschrieben, die mit all Ihren Auto Scaling Scaling-Gruppen verknüpft sind.

```
Get-ASNotificationConfiguration
```

- Einzelheiten zur API finden Sie unter DescribeNotificationConfigurations AWS Tools for PowerShellCmdlet-Referenz.

`Get-ASPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden alle Richtlinien für die angegebene Auto Scaling Scaling-Gruppe beschrieben.

```
Get-ASPolicy -AutoScalingGroupName my-asg
```

Ausgabe:

```
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         : {}
```

Beispiel 2: Dieses Beispiel beschreibt die angegebenen Richtlinien für die angegebene Auto Scaling Scaling-Gruppe.

```
Get-ASPolicy -AutoScalingGroupName my-asg -PolicyName @("myScaleOutPolicy",
 "myScaleInPolicy")
```

Beispiel 3: In diesem Beispiel werden alle Richtlinien für all Ihre Auto Scaling Scaling-Gruppen beschrieben.

```
Get-ASPolicy
```

- Einzelheiten zur API finden Sie unter DescribePolicies AWS Tools for PowerShellCmdlet-Referenz.

`Get-ASScalingActivity`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASScalingActivity`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die Skalierungsaktivitäten der letzten sechs Wochen für die angegebene Auto Scaling Scaling-Gruppe.

```
Get-ASScalingActivity -AutoScalingGroupName my-asg
```

Ausgabe:

```
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        :
```

Beispiel 2: Dieses Beispiel beschreibt die angegebene Skalierungsaktivität.

```
Get-ASScalingActivity -ActivityId "063308ae-aa22-4a9b-94f4-9fae4EXAMPLE"
```

Beispiel 3: Dieses Beispiel beschreibt die Skalierungsaktivitäten der letzten sechs Wochen für all Ihre Auto Scaling Scaling-Gruppen.

```
Get-ASScalingActivity
```

- Einzelheiten zur API finden Sie unter DescribeScalingActivities AWS Tools for PowerShellCmdlet-Referenz.

`Get-ASScalingProcessType`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASScalingProcessType`

Tools für PowerShell

Beispiel 1: Dieses Beispiel listet die Prozesstypen auf, die von Auto Scaling unterstützt werden.

```
Get-ASScalingProcessType
```

Ausgabe:

```
ProcessName
-----------
AZRebalance
AddToLoadBalancer
AlarmNotification
HealthCheck
Launch
ReplaceUnhealthy
ScheduledActions
Terminate
```

- Einzelheiten zur API finden Sie unter DescribeScalingProcessTypes AWS Tools for PowerShellCmdlet-Referenz.

`Get-ASScheduledAction`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASScheduledAction`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die geplanten Skalierungsaktionen für die angegebene Auto Scaling Scaling-Gruppe.

```
Get-ASScheduledAction -AutoScalingGroupName my-asg
```

Ausgabe:

```
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
```

Beispiel 2: Dieses Beispiel beschreibt die angegebenen geplanten Skalierungsaktionen.

```
Get-ASScheduledAction -ScheduledActionName @("myScheduledScaleOut",
 "myScheduledScaleIn")
```

Beispiel 3: Dieses Beispiel beschreibt die geplanten Skalierungsaktionen, die zum angegebenen Zeitpunkt beginnen.

```
Get-ASScheduledAction -StartTime "2015-12-01T08:00:00Z"
```

Beispiel 4: Dieses Beispiel beschreibt die geplanten Skalierungsaktionen, die zum angegebenen Zeitpunkt enden.

```
Get-ASScheduledAction -EndTime "2015-12-30T08:00:00Z"
```

Beispiel 5: In diesem Beispiel werden die geplanten Skalierungsaktionen für alle Ihre Auto Scaling Scaling-Gruppen beschrieben.

```
Get-ASScheduledAction
```

- Einzelheiten zur API finden Sie unter DescribeScheduledActions AWS Tools for PowerShellCmdlet-Referenz.

`Get-ASTag`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASTag`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die Tags mit dem Schlüsselwert „myTag“ oder „myTag2“. Die möglichen Werte für den Filternamen sind '', auto-scaling-group 'key', 'value' und ''. propagate-at-launch Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Get-ASTag -Filter @( @{ Name="key"; Values=@("myTag", "myTag2") } )
```

Ausgabe:

```
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
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um den Filter für den Filter-Parameter zu erstellen.

```
$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 )
```

Beispiel 3: Dieses Beispiel beschreibt alle Tags für all Ihre Auto Scaling Scaling-Gruppen.

```
Get-ASTag
```

- Einzelheiten zur API finden Sie unter DescribeTags AWS Tools for PowerShellCmdlet-Referenz.

`Get-ASTerminationPolicyType`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASTerminationPolicyType`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Kündigungsrichtlinien aufgeführt, die von Auto Scaling unterstützt werden.

```
Get-ASTerminationPolicyType
```

Ausgabe:

```
ClosestToNextInstanceHour
Default
NewestInstance
OldestInstance
OldestLaunchConfiguration
```

- Einzelheiten zur API finden Sie unter DescribeTerminationPolicyTypes AWS Tools for PowerShellCmdlet-Referenz.

`Mount-ASInstance`

Das folgende Codebeispiel zeigt die Verwendung. `Mount-ASInstance`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instance an die angegebene Auto Scaling Scaling-Gruppe angehängt. Auto Scaling erhöht automatisch die gewünschte Kapazität der Auto Scaling Scaling-Gruppe.

```
Mount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter AttachInstances AWS Tools for PowerShellCmdlet-Referenz.

`New-ASAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung. `New-ASAutoScalingGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Auto Scaling Scaling-Gruppe mit dem angegebenen Namen und den angegebenen Attributen erstellt. Die standardmäßig gewünschte Kapazität ist die Mindestgröße. Daher startet diese Auto Scaling Scaling-Gruppe zwei Instances, eine in jeder der angegebenen zwei Availability Zones.

```
New-ASAutoScalingGroup -AutoScalingGroupName my-asg -LaunchConfigurationName my-lc -
MinSize 2 -MaxSize 6 -AvailabilityZone @("us-west-2a", "us-west-2b")
```

- Einzelheiten zur API finden Sie unter CreateAutoScalingGroup AWS Tools for PowerShellCmdlet-Referenz.

`New-ASLaunchConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `New-ASLaunchConfiguration`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Startkonfiguration mit dem Namen „my-lc“ erstellt. Die von Auto Scaling Scaling-Gruppen gestarteten EC2-Instances, die diese Startkonfiguration verwenden, verwenden den angegebenen Instance-Typ, das angegebene AMI, die Sicherheitsgruppe und die IAM-Rolle.

```
New-ASLaunchConfiguration -LaunchConfigurationName my-lc -InstanceType "m3.medium" -
ImageId "ami-12345678" -SecurityGroup "sg-12345678" -IamInstanceProfile "myIamRole"
```

- Einzelheiten zur API finden Sie unter CreateLaunchConfiguration AWS Tools for PowerShellCmdlet-Referenz.

`Remove-ASAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ASAutoScalingGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Auto Scaling Scaling-Gruppe gelöscht, wenn sie keine laufenden Instances hat. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg
```

Ausgabe:

```
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"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie nicht zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -Force
```

Beispiel 3: In diesem Beispiel wird die angegebene Auto Scaling Scaling-Gruppe gelöscht und alle darin enthaltenen laufenden Instances beendet.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -ForceDelete $true -Force
```

- Einzelheiten zur API finden Sie unter DeleteAutoScalingGroup AWS Tools for PowerShellCmdlet-Referenz.

`Remove-ASLaunchConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ASLaunchConfiguration`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Startkonfiguration gelöscht, wenn sie nicht an eine Auto Scaling Scaling-Gruppe angehängt ist. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird.

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc
```

Ausgabe:

```
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"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie nicht zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird.

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc -Force
```

- Einzelheiten zur API finden Sie unter DeleteLaunchConfiguration AWS Tools for PowerShellCmdlet-Referenz.

`Remove-ASLifecycleHook`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ASLifecycleHook`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Lifecycle-Hook für die angegebene Auto Scaling Scaling-Gruppe gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird.

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook
```

Ausgabe:

```
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"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie nicht zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird.

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook -Force
```

- Einzelheiten zur API finden Sie unter DeleteLifecycleHook AWS Tools for PowerShellCmdlet-Referenz.

`Remove-ASNotificationConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ASNotificationConfiguration`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Benachrichtigungsaktion gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird.

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN
 "arn:aws:sns:us-west-2:123456789012:my-topic"
```

Ausgabe:

```
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"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie nicht zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird.

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN
 "arn:aws:sns:us-west-2:123456789012:my-topic" -Force
```

- Einzelheiten zur API finden Sie unter DeleteNotificationConfiguration AWS Tools for PowerShellCmdlet-Referenz.

`Remove-ASPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ASPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Richtlinie für die angegebene Auto Scaling Scaling-Gruppe gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird.

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy
```

Ausgabe:

```
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"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie nicht zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird.

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy -Force
```

- Einzelheiten zur API finden Sie unter DeletePolicy AWS Tools for PowerShellCmdlet-Referenz.

`Remove-ASScheduledAction`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ASScheduledAction`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene geplante Aktion für die angegebene Auto Scaling Scaling-Gruppe gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird.

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction
 "myScheduledAction"
```

Ausgabe:

```
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"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie nicht zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird.

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction
 "myScheduledAction" -Force
```

- Einzelheiten zur API finden Sie unter DeleteScheduledAction AWS Tools for PowerShellCmdlet-Referenz.

`Remove-ASTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ASTag`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Tag aus der angegebenen Auto Scaling Scaling-Gruppe entfernt. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag" } )
```

Ausgabe:

```
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"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie nicht zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird.

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag" } ) -Force
```

Beispiel 3: Bei Powershell Version 2 müssen Sie New-Object verwenden, um das Tag für den Tag-Parameter zu erstellen.

```
$tag = New-Object Amazon.AutoScaling.Model.Tag
$tag.ResourceType = "auto-scaling-group"
$tag.ResourceId = "my-asg"
$tag.Key = "myTag"
Remove-ASTag -Tag $tag -Force
```

- Einzelheiten zur API finden Sie unter DeleteTagsCmdlet-Referenz.AWS Tools for PowerShell

`Resume-ASProcess`

Das folgende Codebeispiel zeigt die Verwendung. `Resume-ASProcess`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Auto Scaling Scaling-Prozess für die angegebene Auto Scaling Scaling-Gruppe wieder aufgenommen.

```
Resume-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"
```

Beispiel 2: In diesem Beispiel werden alle unterbrochenen Auto Scaling Scaling-Prozesse für die angegebene Auto Scaling Scaling-Gruppe wieder aufgenommen.

```
Resume-ASProcess -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter ResumeProcesses AWS Tools for PowerShellCmdlet-Referenz.

`Set-ASDesiredCapacity`

Das folgende Codebeispiel zeigt die Verwendung. `Set-ASDesiredCapacity`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Größe der angegebenen Auto Scaling Scaling-Gruppe festgelegt.

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2
```

Beispiel 2: Dieses Beispiel legt die Größe der angegebenen Auto Scaling Scaling-Gruppe fest und wartet, bis die Abklingzeit abgeschlossen ist, bevor auf die neue Größe skaliert wird.

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2 -HonorCooldown
 $true
```

- Einzelheiten zur API finden Sie unter SetDesiredCapacity AWS Tools for PowerShellCmdlet-Referenz.

`Set-ASInstanceHealth`

Das folgende Codebeispiel zeigt die Verwendung. `Set-ASInstanceHealth`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der Status der angegebenen Instanz auf „Ungesund“ gesetzt, wodurch sie außer Betrieb genommen wird. Auto Scaling beendet und ersetzt die Instanz.

```
Set-ASInstanceHealth -HealthStatus Unhealthy -InstanceId i-93633f9b
```

Beispiel 2: In diesem Beispiel wird der Status der angegebenen Instance auf „Healthy“ gesetzt, sodass sie weiterhin in Betrieb bleibt. Eine Übergangsfrist für Integritätsprüfungen für die Auto Scaling Scaling-Gruppe wird nicht eingehalten.

```
Set-ASInstanceHealth -HealthStatus Healthy -InstanceId i-93633f9b -
ShouldRespectGracePeriod $false
```

- Einzelheiten zur API finden Sie unter SetInstanceHealth AWS Tools for PowerShellCmdlet-Referenz.

`Set-ASInstanceProtection`

Das folgende Codebeispiel zeigt die Verwendung. `Set-ASInstanceProtection`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der Instanzschutz für die angegebene Instanz aktiviert.

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -
ProtectedFromScaleIn $true
```

Beispiel 2: In diesem Beispiel wird der Instanzschutz für die angegebene Instanz deaktiviert.

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -
ProtectedFromScaleIn $false
```

- Einzelheiten zur API finden Sie unter SetInstanceProtection AWS Tools for PowerShellCmdlet-Referenz.

`Set-ASTag`

Das folgende Codebeispiel zeigt die Verwendung. `Set-ASTag`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der angegebenen Auto Scaling Scaling-Gruppe ein einzelnes Tag hinzugefügt. Der Tag-Schlüssel ist 'myTag' und der Tag-Wert ist 'myTagValue'. Auto Scaling leitet dieses Tag an die nachfolgenden EC2-Instances weiter, die von der Auto Scaling Scaling-Gruppe gestartet wurden. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Set-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag"; Value="myTagValue"; PropagateAtLaunch=$true} )
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um das Tag für den Tag-Parameter zu erstellen.

```
$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
```

- Einzelheiten zur API finden Sie unter CreateOrUpdateTags AWS Tools for PowerShellCmdlet-Referenz.

`Start-ASPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Start-ASPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Richtlinie für die angegebene Auto Scaling Scaling-Gruppe ausgeführt.

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy"
```

Beispiel 2: In diesem Beispiel wird die angegebene Richtlinie für die angegebene Auto Scaling Scaling-Gruppe ausgeführt, nachdem auf den Abschluss der Abklingzeit gewartet wurde.

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy" -
HonorCooldown $true
```

- Einzelheiten zur API finden Sie unter ExecutePolicy AWS Tools for PowerShellCmdlet-Referenz.

`Stop-ASInstanceInAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-ASInstanceInAutoScalingGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instance beendet und die gewünschte Kapazität ihrer Auto Scaling-Gruppe verringert, sodass Auto Scaling keine Ersatz-Instance startet.

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $true
```

Ausgabe:

```
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        :
```

Beispiel 2: In diesem Beispiel wird die angegebene Instance beendet, ohne die gewünschte Kapazität ihrer Auto Scaling Scaling-Gruppe zu verringern. Auto Scaling startet eine Ersatzinstanz.

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $false
```

Ausgabe:

```
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        :
```

- Einzelheiten zur API finden Sie unter TerminateInstanceInAutoScalingGroup AWS Tools for PowerShellCmdlet-Referenz.

`Suspend-ASProcess`

Das folgende Codebeispiel zeigt die Verwendung. `Suspend-ASProcess`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Auto Scaling Scaling-Prozess für die angegebene Auto Scaling Scaling-Gruppe unterbrochen.

```
Suspend-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"
```

Beispiel 2: In diesem Beispiel werden alle Auto Scaling Scaling-Prozesse für die angegebene Auto Scaling Scaling-Gruppe unterbrochen.

```
Suspend-ASProcess -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter SuspendProcesses AWS Tools for PowerShellCmdlet-Referenz.

`Update-ASAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Update-ASAutoScalingGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Mindest- und Höchstgröße der angegebenen Auto Scaling Scaling-Gruppe aktualisiert.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -MaxSize 5 -MinSize 1
```

Beispiel 2: In diesem Beispiel wird die Standard-Abklingzeit der angegebenen Auto Scaling Scaling-Gruppe aktualisiert.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -DefaultCooldown 10
```

Beispiel 3: In diesem Beispiel werden die Availability Zones der angegebenen Auto Scaling Scaling-Gruppe aktualisiert.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -AvailabilityZone @("us-
west-2a", "us-west-2b")
```

Beispiel 4: In diesem Beispiel wird die angegebene Auto Scaling Scaling-Gruppe aktualisiert, sodass sie Elastic Load Balancing Health Checks verwendet.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -HealthCheckType ELB -
HealthCheckGracePeriod 60
```

- Einzelheiten zur API finden Sie unter UpdateAutoScalingGroup AWS Tools for PowerShellCmdlet-Referenz.

`Write-ASLifecycleActionHeartbeat`

Das folgende Codebeispiel zeigt die Verwendung. `Write-ASLifecycleActionHeartbeat`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Heartbeat für die angegebene Lebenszyklusaktion aufgezeichnet. Dadurch bleibt die Instanz im Status „Ausstehend“, bis Sie die benutzerdefinierte Aktion abgeschlossen haben.

```
Write-ASLifecycleActionHeartbeat -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook -LifecycleActionToken bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- Einzelheiten zur API finden Sie unter RecordLifecycleActionHeartbeat AWS Tools for PowerShellCmdlet-Referenz.

`Write-ASLifecycleHook`

Das folgende Codebeispiel zeigt die Verwendung. `Write-ASLifecycleHook`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Lifecycle-Hook zur angegebenen Auto Scaling Scaling-Gruppe hinzugefügt.

```
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"
```

- Einzelheiten zur API finden Sie unter PutLifecycleHook AWS Tools for PowerShellCmdlet-Referenz.

`Write-ASNotificationConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Write-ASNotificationConfiguration`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Auto Scaling Scaling-Gruppe so konfiguriert, dass sie beim Starten von EC2-Instances eine Benachrichtigung an das angegebene SNS-Thema sendet.

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType
 "autoscaling:EC2_INSTANCE_LAUNCH" -TopicARN "arn:aws:sns:us-west-2:123456789012:my-
topic"
```

Beispiel 2: In diesem Beispiel wird die angegebene Auto Scaling Scaling-Gruppe so konfiguriert, dass sie beim Starten oder Beenden von EC2-Instances eine Benachrichtigung an das angegebene SNS-Thema sendet.

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType
 @("autoscaling:EC2_INSTANCE_LAUNCH", "autoscaling:EC2_INSTANCE_TERMINATE") -
TopicARN "arn:aws:sns:us-west-2:123456789012:my-topic"
```

- Einzelheiten zur API finden Sie unter PutNotificationConfigurationCmdlet-Referenz.AWS Tools for PowerShell

`Write-ASScalingPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Write-ASScalingPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Richtlinie der angegebenen Auto Scaling Scaling-Gruppe hinzugefügt. Der angegebene Anpassungstyp bestimmt, wie der ScalingAdjustment Parameter interpretiert wird. Bei 'ChangeInCapacity' erhöht ein positiver Wert die Kapazität um die angegebene Anzahl von Instanzen und ein negativer Wert verringert die Kapazität um die angegebene Anzahl von Instanzen.

```
Write-ASScalingPolicy -AutoScalingGroupName my-asg -AdjustmentType
 "ChangeInCapacity" -PolicyName "myScaleInPolicy" -ScalingAdjustment -1
```

Ausgabe:

```
arn:aws:autoscaling:us-west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-
e1d769fc24ef:autoScalingGroupName/my-asg
:policyName/myScaleInPolicy
```

- Einzelheiten zur API finden Sie unter PutScalingPolicy AWS Tools for PowerShellCmdlet-Referenz.

`Write-ASScheduledUpdateGroupAction`

Das folgende Codebeispiel zeigt die Verwendung. `Write-ASScheduledUpdateGroupAction`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine einmalig geplante Aktion erstellt oder aktualisiert, um die gewünschte Kapazität zur angegebenen Startzeit zu ändern.

```
Write-ASScheduledUpdateGroupAction -AutoScalingGroupName my-asg -ScheduledActionName
 "myScheduledAction" -StartTime "2015-12-01T00:00:00Z" -DesiredCapacity 10
```

- Einzelheiten zur API finden Sie unter PutScheduledUpdateGroupAction AWS Tools for PowerShellCmdlet-Referenz.

AWS Budgets Beispiele für die Verwendung von Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS Tools for PowerShell with Aktionen ausführen und allgemeine Szenarien implementieren AWS Budgets.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

Aktionen

`New-BGTBudget`

Das folgende Codebeispiel zeigt, wie Sie es verwenden`New-BGTBudget`.

Tools für PowerShell

Beispiel 1: Erstellt ein neues Budget mit den angegebenen Budget- und Zeitbeschränkungen mit E-Mail-Benachrichtigungen.

```
$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
```

- Einzelheiten zur API finden Sie unter CreateBudget AWS Tools for PowerShellCmdlet-Referenz.

AWS Cloud9 Beispiele für die Verwendung von Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS Tools for PowerShell with Aktionen ausführen und allgemeine Szenarien implementieren AWS Cloud9.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

Aktionen

`Get-C9EnvironmentData`

Das folgende Codebeispiel zeigt, wie Sie es verwenden`Get-C9EnvironmentData`.

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Informationen zu den angegebenen AWS Cloud9-Entwicklungsumgebungen abgerufen.

```
Get-C9EnvironmentData -EnvironmentId
 685f892f431b45c2b28cb69eadcdb0EX,1980b80e5f584920801c09086667f0EX
```

Ausgabe:

```
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
```

Beispiel 2: In diesem Beispiel werden Informationen über den Lebenszyklusstatus der angegebenen AWS Cloud9-Entwicklungsumgebung abgerufen.

```
(Get-C9EnvironmentData -EnvironmentId 685f892f431b45c2b28cb69eadcdb0EX).Lifecycle
```

Ausgabe:

```
FailureResource Reason Status
--------------- ------ ------
                       CREATED
```

- Einzelheiten zur API finden Sie unter DescribeEnvironments AWS Tools for PowerShellCmdlet-Referenz.

`Get-C9EnvironmentList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-C9EnvironmentList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste der verfügbaren Identifikatoren für die AWS Cloud9-Entwicklungsumgebung abgerufen.

```
Get-C9EnvironmentList
```

Ausgabe:

```
685f892f431b45c2b28cb69eadcdb0EX
```

```
1980b80e5f584920801c09086667f0EX
```

- Einzelheiten zur API finden Sie unter ListEnvironments AWS Tools for PowerShellCmdlet-Referenz.

`Get-C9EnvironmentMembershipList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-C9EnvironmentMembershipList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Informationen über Umgebungsmitglieder für die angegebene AWS Cloud9-Entwicklungsumgebung abgerufen.

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

Ausgabe:

```
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
```

Beispiel 2: In diesem Beispiel werden Informationen über den Besitzer der angegebenen AWS Cloud9-Entwicklungsumgebung abgerufen.

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -
Permission owner
```

Ausgabe:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
```

```
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

Beispiel 3: In diesem Beispiel werden Informationen über das angegebene Umgebungsmitglied für mehrere AWS Cloud9-Entwicklungsumgebungen abgerufen.

```
Get-C9EnvironmentMembershipList -UserArn arn:aws:iam::123456789012:user/MyDemoUser
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeEnvironmentMemberships AWS Tools for PowerShellCmdlet-Referenz.

`Get-C9EnvironmentStatus`

Das folgende Codebeispiel zeigt die Verwendung. `Get-C9EnvironmentStatus`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Statusinformationen für die angegebene AWS Cloud9-Entwicklungsumgebung abgerufen.

```
Get-C9EnvironmentStatus -EnvironmentId 349c86d4579e4e7298d500ff57a6b2EX
```

Ausgabe:

```
Message                    Status
```

```
-------                     ------
Environment is ready to use ready
```

- Einzelheiten zur API finden Sie unter DescribeEnvironmentStatus AWS Tools for PowerShellCmdlet-Referenz.

`New-C9EnvironmentEC2`

Das folgende Codebeispiel zeigt die Verwendung. `New-C9EnvironmentEC2`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine AWS Cloud9-Entwicklungsumgebung mit den angegebenen Einstellungen erstellt, eine Amazon Elastic Compute Cloud (Amazon EC2) - Instance gestartet und dann eine Verbindung von der Instance zur Umgebung hergestellt.

```
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
```

Ausgabe:

```
ffd88420d4824eeeaeaa8a04bfde8cEX
```

- Einzelheiten zur API finden Sie unter CreateEnvironmentEc2 in der AWS Tools for PowerShell Cmdlet-Referenz.

`New-C9EnvironmentMembership`

Das folgende Codebeispiel zeigt die Verwendung. `New-C9EnvironmentMembership`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Umgebungsmitglied zur angegebenen AWS Cloud9-Entwicklungsumgebung hinzugefügt.

```
New-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/AnotherDemoUser
 -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Permission read-write
```

Ausgabe:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-write
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX
```

- Einzelheiten zur API finden Sie unter CreateEnvironmentMembership AWS Tools for PowerShellCmdlet-Referenz.

`Remove-C9Environment`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-C9Environment`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene AWS Cloud9-Entwicklungsumgebung gelöscht. Wenn eine Amazon EC2 EC2-Instance mit der Umgebung verbunden ist, wird auch die Instance beendet.

```
Remove-C9Environment -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

- Einzelheiten zur API finden Sie unter DeleteEnvironment AWS Tools for PowerShellCmdlet-Referenz.

`Remove-C9EnvironmentMembership`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-C9EnvironmentMembership`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Umgebungsmitglied aus der angegebenen AWS Cloud9-Entwicklungsumgebung gelöscht.

```
Remove-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

- Einzelheiten zur API finden Sie unter DeleteEnvironmentMembership AWS Tools for PowerShellCmdlet-Referenz.

`Update-C9Environment`

Das folgende Codebeispiel zeigt die Verwendung. `Update-C9Environment`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die angegebenen Einstellungen der angegebenen vorhandenen AWS Cloud9-Entwicklungsumgebung geändert.

```
Update-C9Environment -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Description
 "My changed demonstration development environment." -Name my-changed-demo-env
```

- Einzelheiten zur API finden Sie unter UpdateEnvironment AWS Tools for PowerShellCmdlet-Referenz.

`Update-C9EnvironmentMembership`

Das folgende Codebeispiel zeigt die Verwendung. `Update-C9EnvironmentMembership`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Einstellungen des angegebenen vorhandenen Umgebungsmitglieds für die angegebene AWS Cloud9-Entwicklungsumgebung geändert.

```
Update-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Permission read-
only
```

Ausgabe:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-only
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX
```

- Einzelheiten zur API finden Sie unter UpdateEnvironmentMembership AWS Tools for PowerShellCmdlet-Referenz.

AWS CloudFormation Beispiele für die Verwendung von Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS Tools for PowerShell with Aktionen ausführen und allgemeine Szenarien implementieren AWS CloudFormation.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

Aktionen

`Get-CFNStack`

Das folgende Codebeispiel zeigt, wie Sie es verwenden`Get-CFNStack`.

Tools für PowerShell

Beispiel 1: Gibt eine Sammlung von Stack-Instanzen zurück, die alle Stacks des Benutzers beschreiben.

```
Get-CFNStack
```

Beispiel 2: Gibt eine Stack-Instanz zurück, die den angegebenen Stack beschreibt

```
Get-CFNStack -StackName "myStack"
```

Beispiel 3: Gibt eine Sammlung von Stack-Instanzen zurück, die alle Stacks des Benutzers mithilfe von manuellem Paging beschreiben. Das Starttoken für die nächste Seite wird nach jedem

Aufruf abgerufen, wobei $null bedeutet, dass keine weiteren Details mehr abgerufen werden müssen.

```
$nextToken = $null
do {
    Get-CFNStack -NextToken $nextToken
    $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Einzelheiten zur API finden Sie unter DescribeStacks AWS Tools for PowerShellCmdlet-Referenz.

`Get-CFNStackEvent`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFNStackEvent`

Tools für PowerShell

Beispiel 1: Gibt alle stapelbezogenen Ereignisse für den angegebenen Stack zurück.

```
Get-CFNStackEvent -StackName "myStack"
```

Beispiel 2: Gibt alle stapelbezogenen Ereignisse für den angegebenen Stack zurück, wobei manuelles Paging ab dem angegebenen Token verwendet wird. Das Starttoken für die nächste Seite wird nach jedem Aufruf abgerufen, wobei $null bedeutet, dass keine Ereignisse mehr abgerufen werden müssen.

```
$nextToken = $null
do {
    Get-CFNStack -StackName "myStack" -NextToken $nextToken
    $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Einzelheiten zur API finden Sie unter DescribeStackEvents AWS Tools for PowerShellCmdlet-Referenz.

`Get-CFNStackResource`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFNStackResource`

Tools für PowerShell

Beispiel 1: Gibt die Beschreibung einer Ressource zurück, die in der Vorlage identifiziert wurde, die dem angegebenen Stack durch die logische ID „MyDBInstance“ zugeordnet ist.

```
Get-CFNStackResource -StackName "myStack" -LogicalResourceId "MyDBInstance"
```

- Einzelheiten zur API finden Sie unter DescribeStackResource AWS Tools for PowerShellCmdlet-Referenz.

`Get-CFNStackResourceList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFNStackResourceList`

Tools für PowerShell

Beispiel 1: Gibt die AWS Ressourcenbeschreibungen für bis zu 100 Ressourcen zurück, die dem angegebenen Stack zugeordnet sind. Um Details zu allen Ressourcen zu erhalten, die einem Stack zugeordnet sind, verwenden Sie Get-CFNStackResourceSummary, das auch manuelles Paging der Ergebnisse unterstützt.

```
Get-CFNStackResourceList -StackName "myStack"
```

Beispiel 2: Gibt die Beschreibung der Amazon EC2 EC2-Instance zurück, die in der Vorlage identifiziert wurde, die dem angegebenen Stack durch die logische ID „Ec2Instance“ zugeordnet ist.

```
Get-CFNStackResourceList -StackName "myStack" -LogicalResourceId "Ec2Instance"
```

Beispiel 3: Gibt die Beschreibung von bis zu 100 Ressourcen zurück, die dem Stack zugeordnet sind, der eine Amazon EC2 EC2-Instance enthält, die durch die Instance-ID „i-123456“ identifiziert wird. Um Details zu allen Ressourcen zu erhalten, die einem Stack zugeordnet sind, verwenden Sie Get-CFNStackResourceSummary, das auch manuelles Paging der Ergebnisse unterstützt.

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456"
```

Beispiel 4: Gibt die Beschreibung der Amazon EC2 EC2-Instance zurück, die durch die logische ID „Ec2Instance“ in der Vorlage für einen Stack identifiziert wird. Der Stack wird anhand der

physischen Ressourcen-ID einer darin enthaltenen Ressource identifiziert, in diesem Fall auch einer Amazon EC2 EC2-Instance mit der Instance-ID „i-123456“. Abhängig vom Inhalt der Vorlage könnte auch eine andere physische Ressource verwendet werden, um den Stapel zu identifizieren, z. B. ein Amazon S3 S3-Bucket.

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456" -LogicalResourceId
 "Ec2Instance"
```

- Einzelheiten zur API finden Sie unter DescribeStackResources AWS Tools for PowerShellCmdlet-Referenz.

`Get-CFNStackResourceSummary`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFNStackResourceSummary`

Tools für PowerShell

Beispiel 1: Gibt Beschreibungen aller Ressourcen zurück, die dem angegebenen Stack zugeordnet sind.

```
Get-CFNStackResourceSummary -StackName "myStack"
```

Beispiel 2: Gibt Beschreibungen aller Ressourcen zurück, die dem angegebenen Stack zugeordnet sind, wobei die Ergebnisse manuell durchsucht werden. Das Starttoken für die nächste Seite wird nach jedem Aufruf abgerufen, wobei $null bedeutet, dass keine weiteren Details mehr abgerufen werden müssen.

```
$nextToken = $null
do {
  Get-CFNStackResourceSummary -StackName "myStack" -NextToken $nextToken
  $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Einzelheiten zur API finden Sie unter ListStackResources AWS Tools for PowerShellCmdlet-Referenz.

`Get-CFNStackSummary`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFNStackSummary`

Tools für PowerShell

Beispiel 1: Gibt zusammenfassende Informationen für alle Stapel zurück.

```
Get-CFNStackSummary
```

Beispiel 2: Gibt zusammenfassende Informationen für alle Stapel zurück, die gerade erstellt werden.

```
Get-CFNStackSummary -StackStatusFilter "CREATE_IN_PROGRESS"
```

Beispiel 3: Gibt zusammenfassende Informationen für alle Stapel zurück, die gerade erstellt oder aktualisiert werden.

```
Get-CFNStackSummary -StackStatusFilter @("CREATE_IN_PROGRESS", "UPDATE_IN_PROGRESS")
```

Beispiel 4: Gibt zusammenfassende Informationen für alle Stapel zurück, die derzeit erstellt oder aktualisiert werden, wobei die Ergebnisse manuell durchsucht werden. Das Starttoken für die nächste Seite wird nach jedem Aufruf abgerufen, wobei $null bedeutet, dass keine weiteren Details mehr abgerufen werden müssen.

```
$nextToken = $null
do {
  Get-CFNStackSummary -StackStatusFilter @("CREATE_IN_PROGRESS",
 "UPDATE_IN_PROGRESS") -NextToken $nextToken
  $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Einzelheiten zur API finden Sie unter ListStacks AWS Tools for PowerShellCmdlet-Referenz.

`Get-CFNTemplate`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFNTemplate`

Tools für PowerShell

Beispiel 1: Gibt die Vorlage zurück, die dem angegebenen Stack zugeordnet ist.

```
Get-CFNTemplate -StackName "myStack"
```

- Einzelheiten zur API finden Sie unter GetTemplate AWS Tools for PowerShellCmdlet-Referenz.

`Measure-CFNTemplateCost`

Das folgende Codebeispiel zeigt die Verwendung. `Measure-CFNTemplateCost`

Tools für PowerShell

Beispiel 1: Gibt eine URL für den AWS einfachen Monatsrechner mit einer Abfragezeichenfolge zurück, die die Ressourcen beschreibt, die zum Ausführen der Vorlage erforderlich sind. Die Vorlage wird aus der angegebenen Amazon S3 S3-URL abgerufen und der einzelne angewendete Anpassungsparameter verwendet. Der Parameter kann auch mit 'Key' und 'Value' anstelle von '' und ParameterKey '' angegeben werden. ParameterValue

```
Measure-CFNTemplateCost -TemplateURL https://s3.amazonaws.com/mytemplates/
templatefile.template `
                        -Region us-west-1 `
                        -Parameter @{ ParameterKey="KeyName";
 ParameterValue="myKeyPairName" }
```

Beispiel 2: Gibt eine AWS Simple Monthly Calculator-URL mit einer Abfragezeichenfolge zurück, die die Ressourcen beschreibt, die zum Ausführen der Vorlage erforderlich sind. Die Vorlage wird anhand des bereitgestellten Inhalts analysiert und die angewendeten Anpassungsparameter angewendet (in diesem Beispiel wird davon ausgegangen, dass der Vorlageninhalt zwei Parameter deklariert hätte, 'KeyName' und 'InstanceType'). Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und 'ParameterKey' angegeben werden. ParameterValue

```
Measure-CFNTemplateCost -TemplateBody "{TEMPLATE CONTENT HERE}" `
                        -Parameter @( @{ ParameterKey="KeyName";
 ParameterValue="myKeyPairName" },`
                                      @{ ParameterKey="InstanceType";
 ParameterValue="m1.large" })
```

Beispiel 3: Verwendet New-Object, um den Satz von Vorlagenparametern zu erstellen, und gibt eine AWS Simple Monthly Calculator-URL mit einer Abfragezeichenfolge zurück, die die für die Ausführung der Vorlage erforderlichen Ressourcen beschreibt. Die Vorlage wird anhand des bereitgestellten Inhalts mit Anpassungsparametern analysiert (in diesem Beispiel wird davon ausgegangen, dass der Vorlageninhalt zwei Parameter deklariert hätte, '' und KeyName ''). 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 )
```

- Einzelheiten zur API finden Sie unter EstimateTemplateCost AWS Tools for PowerShellCmdlet-Referenz.

`New-CFNStack`

Das folgende Codebeispiel zeigt die Verwendung. `New-CFNStack`

Tools für PowerShell

Beispiel 1: Erzeugt einen neuen Stack mit dem angegebenen Namen. Die Vorlage wird anhand des bereitgestellten Inhalts mit Anpassungsparametern analysiert ('PK1' und 'PK2' stehen für die Namen der im Vorlageninhalt deklarierten Parameter, 'PV1' und 'PV2' stehen für die Werte für diese Parameter). Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und '' angegeben werden. ParameterKey ParameterValue Wenn die Erstellung des Stacks fehlschlägt, wird er nicht zurückgesetzt.

```
New-CFNStack -StackName "myStack" `
             -TemplateBody "{TEMPLATE CONTENT HERE}" `
             -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" }) `
             -DisableRollback $true
```

Beispiel 2: Erstellt einen neuen Stack mit dem angegebenen Namen. Die Vorlage wird anhand des bereitgestellten Inhalts mit Anpassungsparametern analysiert ('PK1' und 'PK2' stehen für die Namen der im Vorlageninhalt deklarierten Parameter, 'PV1' und 'PV2' stehen für die Werte für diese Parameter). Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und '' angegeben werden. ParameterKey ParameterValue Wenn die Erstellung des Stacks fehlschlägt, wird er zurückgesetzt.

```
$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"
```

Beispiel 3: Erzeugt einen neuen Stack mit dem angegebenen Namen. Die Vorlage wird von der Amazon S3 S3-URL mit Anpassungsparametern abgerufen ('PK1' steht für den Namen eines Parameters, der im Inhalt der Vorlage deklariert ist, 'PV1' steht für den Wert für den Parameter). Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und 'ParameterKey' angegeben werden. ParameterValue Wenn die Erstellung des Stacks fehlschlägt, wird er zurückgesetzt (genauso wie bei der Angabe von - DisableRollback $false).

```
New-CFNStack -StackName "myStack" `
             -TemplateURL https://s3.amazonaws.com/mytemplates/templatefile.template
 `
             -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Beispiel 4: Erzeugt einen neuen Stack mit dem angegebenen Namen. Die Vorlage wird von der Amazon S3 S3-URL mit Anpassungsparametern abgerufen ('PK1' steht für den Namen eines Parameters, der im Inhalt der Vorlage deklariert ist, 'PV1' steht für den Wert für den Parameter). Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und 'ParameterKey' angegeben werden. ParameterValue Wenn die Erstellung des Stacks fehlschlägt, wird er zurückgesetzt (genauso wie bei der Angabe von - DisableRollback $false). Die angegebene Benachrichtigung: AENs erhalten veröffentlichte Ereignisse im Zusammenhang mit dem Stack.

```
New-CFNStack -StackName "myStack" `
             -TemplateURL https://s3.amazonaws.com/mytemplates/templatefile.template
 `
             -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" } `
             -NotificationARN @( "arn1", "arn2" )
```

- Einzelheiten zur API finden Sie unter CreateStackCmdlet-Referenz.AWS Tools for PowerShell

`Remove-CFNStack`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-CFNStack`

Tools für PowerShell

Beispiel 1: Löscht den angegebenen Stapel.

```
Remove-CFNStack -StackName "myStack"
```

- Einzelheiten zur API finden Sie unter DeleteStack AWS Tools for PowerShellCmdlet-Referenz.

`Resume-CFNUpdateRollback`

Das folgende Codebeispiel zeigt die Verwendung. `Resume-CFNUpdateRollback`

Tools für PowerShell

Beispiel 1: Setzt das Rollback des benannten Stacks fort, der sich im Status 'UPDATE_ROLLBACK_FAILED' befinden sollte. Wenn das fortgesetzte Rollback erfolgreich ist, wechselt der Stack in den Status „UPDATE_ROLLBACK_COMPLETE“.

```
Resume-CFNUpdateRollback -StackName "myStack"
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz. ContinueUpdateRollbackAWS Tools for PowerShell

`Stop-CFNUpdateStack`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-CFNUpdateStack`

Tools für PowerShell

Beispiel 1: Bricht ein Update auf dem angegebenen Stack ab.

```
Stop-CFNUpdateStack -StackName "myStack"
```

- Einzelheiten zur API finden Sie unter CancelUpdateStack AWS Tools for PowerShellCmdlet-Referenz.

`Test-CFNStack`

Das folgende Codebeispiel zeigt die Verwendung. `Test-CFNStack`

Tools für PowerShell

Beispiel 1: Testet, ob der Stack einen der Zustände UPDATE_ROLLBACK_COMPLETE, CREATE_COMPLETE, ROLLBACK_COMPLETE oder UPDATE_COMPLETE erreicht hat.

```
Test-CFNStack -StackName MyStack
```

Ausgabe:

```
False
```

Beispiel 2: Testet, ob der Stack den Status UPDATE_COMPLETE oder UPDATE_ROLLBACK_COMPLETE erreicht hat.

```
Test-CFNStack -StackName MyStack -Status UPDATE_COMPLETE,UPDATE_ROLLBACK_COMPLETE
```

Ausgabe:

```
True
```

- API-Details finden Sie unter Test-CFNStack in der Cmdlet-Referenz.AWS Tools for PowerShell

`Test-CFNTemplate`

Das folgende Codebeispiel zeigt die Verwendung. `Test-CFNTemplate`

Tools für PowerShell

Beispiel 1: Überprüft den angegebenen Vorlageninhalt. In der Ausgabe werden die Funktionen, die Beschreibung und die Parameter der Vorlage detailliert beschrieben.

```
Test-CFNTemplate -TemplateBody "{TEMPLATE CONTENT HERE}"
```

Beispiel 2: Validiert die angegebene Vorlage, auf die über eine Amazon S3 S3-URL zugegriffen wurde. In der Ausgabe werden die Funktionen, die Beschreibung und die Parameter der Vorlage detailliert beschrieben.

```
Test-CFNTemplate -TemplateURL https://s3.amazonaws.com/mytemplates/
templatefile.template
```

- Einzelheiten zur API finden Sie unter ValidateTemplate AWS Tools for PowerShellCmdlet-Referenz.

`Update-CFNStack`

Das folgende Codebeispiel zeigt die Verwendung. `Update-CFNStack`

Tools für PowerShell

Beispiel 1: Aktualisiert den Stack 'MyStack' mit den angegebenen Vorlagen- und Anpassungsparametern. 'PK1' steht für den Namen eines in der Vorlage deklarierten Parameters und 'PV1' für seinen Wert. Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und 'ParameterKey' angegeben werden. ParameterValue

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Beispiel 2: Aktualisiert den Stack 'MyStack' mit den angegebenen Vorlagen- und Anpassungsparametern. 'PK1' und 'PK2' stehen für die Namen der in der Vorlage deklarierten Parameter, 'PV1' und 'PV2' stehen für ihre angeforderten Werte. Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und '' angegeben werden. ParameterKey ParameterValue

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

Beispiel 3: Aktualisiert den Stack 'MyStack' mit den angegebenen Vorlagen- und Anpassungsparametern. 'PK1' steht für den Namen eines in der Vorlage deklarierten Parameters und 'PV2' für seinen Wert. Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und 'ParameterKey' angegeben werden. ParameterValue

```
Update-CFNStack -StackName "myStack" -TemplateBody "{Template Content Here}" -
Parameters @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Beispiel 4: Aktualisiert den Stack 'MyStack' mit der angegebenen Vorlage, die von Amazon S3 abgerufen wurde, und den Anpassungsparametern. 'PK1' und 'PK2' stehen für die Namen der in der Vorlage deklarierten Parameter, 'PV1' und 'PV2' stehen für ihre angeforderten Werte. Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und '' angegeben werden. ParameterKey ParameterValue

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/mytemplates/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

Beispiel 5: Aktualisiert den Stack 'MyStack', von dem in diesem Beispiel angenommen wird, dass er IAM-Ressourcen enthält, mit der angegebenen Vorlage, die von Amazon S3 abgerufen wurde, und den Anpassungsparametern. 'PK1' und 'PK2' stehen für die Namen der in der Vorlage deklarierten Parameter, 'PV1' und 'PV2' stehen für ihre angeforderten Werte. Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und '' angegeben werden. ParameterKey ParameterValue Bei Stacks, die IAM-Ressourcen enthalten, müssen Sie den Capability-Parameter „CAPABILITY_IAM“ angeben. Andernfalls schlägt das Update mit einem Fehler '' fehl. InsufficientCapabilities

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/mytemplates/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } ) `
                -Capabilities "CAPABILITY_IAM"
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz. UpdateStackAWS Tools for PowerShell

`Wait-CFNStack`

Das folgende Codebeispiel zeigt die Verwendung. `Wait-CFNStack`

Tools für PowerShell

Beispiel 1: Testet, ob der Stack einen der Zustände UPDATE_ROLLBACK_COMPLETE, CREATE_COMPLETE, ROLLBACK_COMPLETE oder UPDATE_COMPLETE erreicht hat. Wenn sich der Stack nicht in einem der Zustände befindet, ruht der Befehl zwei Sekunden lang, bevor er den Status erneut testet. Dies wird wiederholt, bis der Stack einen der angeforderten Zustände

erreicht hat oder die Standard-Timeout-Periode von 60 Sekunden abgelaufen ist. Wenn der Timeout-Zeitraum überschritten wird, wird eine Ausnahme ausgelöst. Wenn der Stack innerhalb des Timeout-Zeitraums einen der angeforderten Zustände erreicht, wird er an die Pipeline zurückgegeben.

```
$stack = Wait-CFNStack -StackName MyStack
```

Beispiel 2: In diesem Beispiel wird insgesamt 5 Minuten (300 Sekunden) gewartet, bis der Stack einen der angegebenen Zustände erreicht. In diesem Beispiel wird der Status vor dem Timeout erreicht und das Stack-Objekt wird daher an die Pipeline zurückgegeben.

```
Wait-CFNStack -StackName MyStack -Timeout 300 -Status
 CREATE_COMPLETE,ROLLBACK_COMPLETE
```

Ausgabe:

```
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
```

Beispiel 3: Dieses Beispiel zeigt die Fehlerausgabe, wenn ein Stack innerhalb des Timeout-Zeitraums (in diesem Fall der Standardzeitraum von 60 Sekunden) keinen der angeforderten Zustände erreicht.

```
Wait-CFNStack -StackName MyStack -Status CREATE_COMPLETE,ROLLBACK_COMPLETE
```

Ausgabe:

```
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
```

- API-Details finden Sie unter Wait-CFNStack in der Cmdlet-Referenz.AWS Tools for PowerShell

CloudFront Beispiele für die Verwendung von Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS Tools for PowerShell with Aktionen ausführen und allgemeine Szenarien implementieren CloudFront.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`Get-CFCloudFrontOriginAccessIdentity`

Das folgende Codebeispiel zeigt, wie Sie es verwendenGet-CFCloudFrontOriginAccessIdentity.

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine bestimmte CloudFront Amazon-Ursprungszugriffsidentität zurückgegeben, die durch den Parameter -Id angegeben wird. Der Parameter -Id ist zwar nicht erforderlich, aber wenn Sie ihn nicht angeben, werden keine Ergebnisse zurückgegeben.

```
Get-CFCloudFrontOriginAccessIdentity -Id E3XXXXXXXXXXRT
```

Ausgabe:

```
    CloudFrontOriginAccessIdentityConfig    Id
S3CanonicalUserId
    ------------------------------------    --
----------------
    Amazon.CloudFront.Model.CloudFrontOr... E3XXXXXXXXXXRT
4b6e...
```

- Einzelheiten zur API finden Sie unter GetCloudFrontOriginAccessIdentity AWS Tools for PowerShellCmdlet-Referenz.

`Get-CFCloudFrontOriginAccessIdentityConfig`

Das folgende Codebeispiel zeigt die Verwendung. Get-CFCloudFrontOriginAccessIdentityConfig

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Konfigurationsinformationen zu einer einzelnen CloudFront Amazon-Ursprungszugriffsidentität zurückgegeben, die durch den Parameter -Id angegeben wird. Fehler treten auf, wenn kein -Id-Parameter angegeben ist.

```
Get-CFCloudFrontOriginAccessIdentityConfig -Id E3XXXXXXXXXXRT
```

Ausgabe:

```
    CallerReference                                          Comment
    ---------------                                          -------
    mycallerreference: 2/1/2011 1:16:32 PM                   Caller reference:
2/1/2011 1:16:32 PM
```

- Einzelheiten zur API finden Sie unter GetCloudFrontOriginAccessIdentityConfig AWS Tools for PowerShellCmdlet-Referenz.

`Get-CFCloudFrontOriginAccessIdentityList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFCloudFrontOriginAccessIdentityList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste der CloudFront Amazon-Origin-Zugriffsidentitäten zurückgegeben. Da der MaxItem Parameter - den Wert 2 angibt, enthalten die Ergebnisse zwei Identitäten.

```
Get-CFCloudFrontOriginAccessIdentityList -MaxItem 2
```

Ausgabe:

```
IsTruncated : True
Items       : {E326XXXXXXXXXT, E1YWXXXXXXX9B}
Marker      :
MaxItems    : 2
NextMarker  : E1YXXXXXXXXX9B
Quantity    : 2
```

- Einzelheiten zur API finden Sie unter ListCloudFrontOriginAccessIdentities AWS Tools for PowerShellCmdlet-Referenz.

`Get-CFDistribution`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFDistribution`

Tools für PowerShell

Beispiel 1: Ruft die Informationen für eine bestimmte Distribution ab.

```
Get-CFDistribution -Id EXAMPLE0000ID
```

- Einzelheiten zur API finden Sie unter GetDistribution AWS Tools for PowerShellCmdlet-Referenz.

`Get-CFDistributionConfig`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFDistributionConfig`

Tools für PowerShell

Beispiel 1: Ruft die Konfiguration für eine bestimmte Distribution ab.

```
Get-CFDistributionConfig -Id EXAMPLE0000ID
```

- Einzelheiten zur API finden Sie unter GetDistributionConfig AWS Tools for PowerShellCmdlet-Referenz.

`Get-CFDistributionList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFDistributionList`

Tools für PowerShell

Beispiel 1: Gibt Verteilungen zurück.

```
Get-CFDistributionList
```

- Einzelheiten zur API finden Sie unter ListDistributions AWS Tools for PowerShellCmdlet-Referenz.

`New-CFDistribution`

Das folgende Codebeispiel zeigt die Verwendung. `New-CFDistribution`

Tools für PowerShell

Beispiel 1: Erstellt eine CloudFront Basisdistribution, die mit Protokollierung und Caching konfiguriert ist.

```
$origin = New-Object Amazon.CloudFront.Model.Origin
$origin.DomainName = "ps-cmdlet-sample.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 ps-cmdlet-sample-logging.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
```

- Einzelheiten zur API finden Sie unter CreateDistribution AWS Tools for PowerShellCmdlet-Referenz.

`New-CFInvalidation`

Das folgende Codebeispiel zeigt die Verwendung. `New-CFInvalidation`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine neue Invalidierung für eine Distribution mit der ID EXAMPLENSTXAXE erstellt. Das CallerReference ist eine eindeutige ID, die vom Benutzer ausgewählt wurde. In diesem Fall wird ein Zeitstempel verwendet, der den 15. Mai 2019 um 9:00 Uhr darstellt. Die Variable $Paths speichert drei Pfade zu Bild- und Mediendateien, die der

Benutzer nicht im Cache der Distribution haben möchte. Der Parameterwert -Paths_Quantity ist die Gesamtzahl der im Parameter -Paths_Item angegebenen Pfade.

```
$Paths = "/images/*.gif", "/images/image1.jpg", "/videos/*.mp4"
New-CFInvalidation -DistributionId "EXAMPLENSTXAXE" -
InvalidationBatch_CallerReference 20190515090000 -Paths_Item $Paths -Paths_Quantity
 3
```

Ausgabe:

```
Invalidation                          Location

------------                          --------

Amazon.CloudFront.Model.Invalidation https://cloudfront.amazonaws.com/2018-11-05/
distribution/EXAMPLENSTXAXE/invalidation/EXAMPLE8NOK9H
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz. CreateInvalidationAWS Tools for PowerShell

`New-CFSignedCookie`

Das folgende Codebeispiel zeigt die Verwendung. `New-CFSignedCookie`

Tools für PowerShell

Beispiel 1: Erstellt mithilfe einer vorgefertigten Richtlinie ein signiertes Cookie für die angegebene Ressource. Das Cookie ist ein Jahr lang gültig.

```
$params = @{
 "ResourceUri"="http://xyz.cloudfront.net/image1.jpeg"
 "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
 "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
 "ExpiresOn"=(Get-Date).AddYears(1)
}
New-CFSignedCookie @params
```

Ausgabe:

```
Expires
```

```
-------
[CloudFront-Expires, 1472227284]
```

Beispiel 2: Erstellt mithilfe einer benutzerdefinierten Richtlinie ein signiertes Cookie für die angegebenen Ressourcen. Das Cookie ist innerhalb von 24 Stunden gültig und läuft eine Woche danach ab.

```
$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
```

Ausgabe:

```
Policy
------
[CloudFront-Policy, eyJTd...wIjo...
```

Beispiel 3: Erstellt mithilfe einer benutzerdefinierten Richtlinie ein signiertes Cookie für die angegebenen Ressourcen. Das Cookie ist innerhalb von 24 Stunden gültig und läuft eine Woche danach ab. Der Zugriff auf die Ressourcen ist auf den angegebenen IP-Bereich beschränkt.

```
$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
```

Ausgabe:

```
Policy
                                                                  ------

                                    [CloudFront-Policy, eyJTd...wIjo...
```

- Einzelheiten zur API finden Sie unter New-CF SignedCookie in der AWS Tools for PowerShell Cmdlet-Referenz.

`New-CFSignedUrl`

Das folgende Codebeispiel zeigt die Verwendung. `New-CFSignedUrl`

Tools für PowerShell

Beispiel 1: Erstellt mithilfe einer vorgefertigten Richtlinie eine signierte URL zur angegebenen Ressource. Die URL wird eine Stunde lang gültig sein. Ein System.Uri-Objekt, das die signierte URL enthält, wird an die Pipeline ausgegeben.

```
$params = @{
 "ResourceUri"="https://cdn.example.com/index.html"
 "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
 "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
 "ExpiresOn"=(Get-Date).AddHours(1)
}
New-CFSignedUrl @params
```

Beispiel 2: Erstellt mithilfe einer benutzerdefinierten Richtlinie eine signierte URL für die angegebene Ressource. Die URL ist ab 24 Stunden gültig und läuft eine Woche später ab.

```
$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
```

Beispiel 3: Erstellt mithilfe einer benutzerdefinierten Richtlinie eine signierte URL zu der angegebenen Ressource. Die URL ist ab 24 Stunden gültig und läuft eine Woche später ab. Der Zugriff auf die Ressource ist auf den angegebenen IP-Bereich beschränkt.

```
$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
```

- Einzelheiten zur API finden Sie unter New-CF SignedUrl in der AWS Tools for PowerShell Cmdlet-Referenz.

CloudTrail Beispiele für die Verwendung von Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS Tools for PowerShell with Aktionen ausführen und allgemeine Szenarien implementieren CloudTrail.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`Find-CTEvent`

Das folgende Codebeispiel zeigt, wie Sie es verwenden`Find-CTEvent`.

Tools für PowerShell

Beispiel 1: Gibt alle Ereignisse zurück, die in den letzten sieben Tagen aufgetreten sind. Das Cmdlet führt standardmäßig automatisch mehrere Aufrufe durch, um alle Ereignisse zu übermitteln. Es wird beendet, wenn der Dienst anzeigt, dass keine weiteren Daten verfügbar sind.

```
Find-CTEvent
```

Beispiel 2: Gibt alle Ereignisse zurück, die in den letzten sieben Tagen aufgetreten sind, und gibt eine Region an, die nicht der aktuelle Shell-Standard ist.

```
Find-CTEvent -Region eu-central-1
```

Beispiel 3: Gibt alle Ereignisse zurück, die mit dem RunInstances API-Aufruf verknüpft sind.

```
Find-CTEvent -LookupAttribute @{ AttributeKey="EventName";
 AttributeValue="RunInstances" }
```

Beispiel 4: Gibt die ersten 5 verfügbaren Ereignisse zurück. Das Token, das zum Abrufen weiterer Ereignisse verwendet werden soll, wird dem **`$AWSHistory.LastServiceResponse`** Mitglied als Notizeigenschaft mit dem Namen NextToken " angehängt.

```
Find-CTEvent -MaxResult 5
```

Beispiel 5: Gibt die nächsten 10 Ereignisse zurück, wobei das Token „nächste Seite“ aus einem vorherigen Aufruf verwendet wird, um anzugeben, von wo in der Sequenz mit der Rückgabe von Ereignissen begonnen werden soll.

```
Find-CTEvent -MaxResult 10 -NextToken $AWSHistory.LastServiceResponse.NextToken
```

Beispiel 6: Dieses Beispiel zeigt, wie die verfügbaren Ereignisse mithilfe von manuellem Paging in einer Schleife durchsucht werden, wobei maximal 5 Ereignisse pro Anruf abgerufen werden.

```
$nextToken = $null
do
{
    Find-CTEvent -MaxResult 5 -NextToken $nextToken
    $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Einzelheiten zur API finden Sie unter LookupEvents AWS Tools for PowerShellCmdlet-Referenz.

`Get-CTTrail`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CTTrail`

Tools für PowerShell

Beispiel 1: Gibt die Einstellungen aller Trails zurück, die mit der aktuellen Region für Ihr Konto verknüpft sind.

```
Get-CTTrail
```

Beispiel 2: Gibt die Einstellungen für die angegebenen Wanderwege zurück.

```
Get-CTTrail -TrailNameList trail1,trail2
```

Beispiel 3: Gibt die Einstellungen für die angegebenen Pfade zurück, die in einer anderen Region als dem aktuellen Shell-Standard erstellt wurden (in diesem Fall der Region Frankfurt (eu-central-1)).

```
Get-CTTrail -TrailNameList trailABC,trailDEF -Region eu-central-1
```

- Einzelheiten zur API finden Sie unter DescribeTrailsCmdlet-Referenz.AWS Tools for PowerShell

`Get-CTTrailStatus`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CTTrailStatus`

Tools für PowerShell

Beispiel 1: Gibt Statusinformationen für den Trail mit dem Namen 'myExampleTrail' zurück. Zu den zurückgegebenen Daten gehören Informationen zu Lieferfehlern, Amazon SNS- und Amazon S3 S3-Fehlern sowie zu den Start- und Endzeiten der Protokollierung für den Trail. In diesem Beispiel wird davon ausgegangen, dass der Trail in derselben Region wie der aktuelle Shell-Standard erstellt wurde.

```
Get-CTTrailStatus -Name myExampleTrail
```

Beispiel 2: Gibt Statusinformationen für einen Trail zurück, der in einer anderen Region als dem aktuellen Shell-Standard erstellt wurde (in diesem Fall der Region Frankfurt (eu-central-1)).

```
Get-CTTrailStatus -Name myExampleTrail -Region eu-central-1
```

- Einzelheiten zur API finden Sie unter GetTrailStatusCmdlet-Referenz.AWS Tools for PowerShell

`New-CTTrail`

Das folgende Codebeispiel zeigt die Verwendung. `New-CTTrail`

Tools für PowerShell

Beispiel 1: Erstellt einen Trail, der den Bucket 'mycloudtrailbucket' für die Speicherung von Protokolldateien verwendet.

```
New-CTTrail -Name="awscloudtrail-example" -S3BucketName="mycloudtrailbucket"
```

Beispiel 2: Erstellt einen Trail, der den Bucket 'mycloudtrailbucket' für die Speicherung von Protokolldateien verwendet. Die S3-Objekte, die die Protokolle darstellen, werden das gemeinsame key prefix „mylogs“ haben. Wenn neue Protokolle an den Bucket übermittelt werden, wird eine Benachrichtigung an das SNS-Thema „mlog-deliverytopic“ gesendet. In diesem Beispiel wird Splatting verwendet, um die Parameterwerte für das Cmdlet bereitzustellen.

```
$params = @{
    Name="awscloudtrail-example"
    S3BucketName="mycloudtrailbucket"
    S3KeyPrefix="mylogs"
```

```
    SnsTopicName="mlog-deliverytopic"
}
New-CTTrail @params
```

- Einzelheiten zur API finden Sie unter CreateTrailCmdlet-Referenz.AWS Tools for PowerShell

`Remove-CTTrail`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-CTTrail`

Tools für PowerShell

Beispiel 1: Löscht den angegebenen Trail. Sie werden zur Bestätigung aufgefordert, bevor der Befehl ausgeführt wird. Um die Bestätigung zu unterdrücken, fügen Sie den Switch-Parameter -Force hinzu.

```
Remove-CTTrail -Name "awscloudtrail-example"
```

- Einzelheiten zur API finden Sie unter DeleteTrail AWS Tools for PowerShellCmdlet-Referenz.

`Start-CTLogging`

Das folgende Codebeispiel zeigt die Verwendung. `Start-CTLogging`

Tools für PowerShell

Beispiel 1: Startet die Aufzeichnung von AWS API-Aufrufen und die Bereitstellung von Protokolldateien für den Trail mit dem Namen 'myExampleTrail'. In diesem Beispiel wird davon ausgegangen, dass der Trail in derselben Region wie der aktuelle Shell-Standard erstellt wurde.

```
Start-CTLogging -Name myExampleTrail
```

Beispiel 2: Startet die Aufzeichnung von AWS API-Aufrufen und die Bereitstellung von Protokolldateien für einen Trail, der in einer anderen Region als dem aktuellen Shell-Standard erstellt wurde (in diesem Fall der Region Frankfurt (eu-central-1)).

```
Start-CTLogging -Name myExampleTrail -Region eu-central-1
```

- Einzelheiten zur API finden Sie unter StartLoggingCmdlet-Referenz.AWS Tools for PowerShell

`Stop-CTLogging`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-CTLogging`

Tools für PowerShell

Beispiel 1: Unterbricht die Aufzeichnung von AWS API-Aufrufen und die Bereitstellung von Protokolldateien für den Trail mit dem Namen 'myExampleTrail'. In diesem Beispiel wird davon ausgegangen, dass der Trail in derselben Region wie der aktuelle Shell-Standard erstellt wurde.

```
Stop-CTLogging -Name myExampleTrail
```

Beispiel 2: Unterbricht die Aufzeichnung von AWS API-Aufrufen und die Bereitstellung von Protokolldateien für einen Trail, der in einer anderen Region als der aktuellen Shell-Standardregion erstellt wurde (in diesem Fall in der Region Frankfurt (eu-central-1)).

```
Stop-CTLogging -Name myExampleTrail -Region eu-central-1
```

- Einzelheiten zur API finden Sie unter StopLoggingCmdlet-Referenz.AWS Tools for PowerShell

`Update-CTTrail`

Das folgende Codebeispiel zeigt die Verwendung. `Update-CTTrail`

Tools für PowerShell

Beispiel 1: Aktualisiert den angegebenen Trail, sodass globale Serviceereignisse (z. B. von IAM) aufgezeichnet werden, und ändert das gemeinsame key prefix der Protokolldateien in Zukunft in „Globallogs“.

```
Update-CTTrail -Name "awscloudtrail-example" -IncludeGlobalServiceEvents $true -
S3KeyPrefix "globallogs"
```

Beispiel 2: Aktualisiert den angegebenen Pfad, sodass Benachrichtigungen über neue Protokollzustellungen an das angegebene SNS-Thema gesendet werden.

```
Update-CTTrail -Name "awscloudtrail-example" -SnsTopicName "mlog-deliverytopic2"
```

Beispiel 3: Aktualisiert den angegebenen Pfad, sodass die Protokolle an einen anderen Bucket gesendet werden.

```
Update-CTTrail -Name "awscloudtrail-example" -S3BucketName "otherlogs"
```

- Einzelheiten zur API finden Sie unter UpdateTrail AWS Tools for PowerShellCmdlet-Referenz.

CloudWatch Beispiele für die Verwendung von Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS Tools for PowerShell with Aktionen ausführen und allgemeine Szenarien implementieren CloudWatch.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`Get-CWDashboard`

Das folgende Codebeispiel zeigt, wie Sie es verwenden`Get-CWDashboard`.

Tools für PowerShell

Beispiel 1: Gibt den Hauptteil des angegebenen Dashboards zurück.

```
Get-CWDashboard -DashboardName Dashboard1
```

Ausgabe:

```
DashboardArn                                          DashboardBody
------------                                          -------------
arn:aws:cloudwatch::123456789012:dashboard/Dashboard1 {...
```

- Einzelheiten zur API finden Sie unter GetDashboard AWS Tools for PowerShellCmdlet-Referenz.

`Get-CWDashboardList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CWDashboardList`

Tools für PowerShell

Beispiel 1: Gibt die Sammlung von Dashboards für Ihr Konto zurück.

```
Get-CWDashboardList
```

Ausgabe:

```
DashboardArn DashboardName LastModified        Size
------------ ------------- ------------        ----
arn:...      Dashboard1    7/6/2017 8:14:15 PM 252
```

Beispiel 2: Gibt die Sammlung von Dashboards für Ihr Konto zurück, deren Namen mit dem Präfix „dev“ beginnen.

```
Get-CWDashboardList -DashboardNamePrefix dev
```

- Einzelheiten zur API finden Sie unter ListDashboardsCmdlet-Referenz.AWS Tools for PowerShell

`Remove-CWDashboard`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-CWDashboard`

Tools für PowerShell

Beispiel 1: Löscht das angegebene Dashboard und lädt zur Bestätigung ein, bevor der Vorgang fortgesetzt wird. Um die Bestätigung zu umgehen, fügen Sie dem Befehl den Schalter -Force hinzu.

```
Remove-CWDashboard -DashboardName Dashboard1
```

- Einzelheiten zur API finden Sie unter DeleteDashboards AWS Tools for PowerShellCmdlet-Referenz.

`Write-CWDashboard`

Das folgende Codebeispiel zeigt die Verwendung. `Write-CWDashboard`

Tools für PowerShell

Beispiel 1: Erstellt oder aktualisiert das Dashboard mit dem Namen 'Dashboard1', sodass es zwei Metrik-Widgets nebeneinander enthält.

```
$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",
                      "MyBucketName"
                   ]
                ],
                "period":86400,
                "stat":"Maximum",
                "region":"us-east-1",
                "title":"MyBucketName bytes"
            }
        }
    ]
}
"@

Write-CWDashboard -DashboardName Dashboard1 -DashboardBody $dashBody
```

Beispiel 2: Erstellt oder aktualisiert das Dashboard und leitet den Inhalt, der das Dashboard beschreibt, über die Pipeline an das Cmdlet weiter.

```
$dashBody = @"
{
...
}
"@

$dashBody | Write-CWDashboard -DashboardName Dashboard1
```

- Einzelheiten zur API finden Sie unter PutDashboardCmdlet-Referenz.AWS Tools for PowerShell

`Write-CWMetricData`

Das folgende Codebeispiel zeigt die Verwendung. `Write-CWMetricData`

Tools für PowerShell

Beispiel 1: Erstellt ein neues MetricDatum Objekt und schreibt es in 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
```

- Einzelheiten zur API finden Sie unter PutMetricData AWS Tools for PowerShellCmdlet-Referenz.

CodeCommit Beispiele für die Verwendung von Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS Tools for PowerShell with Aktionen ausführen und allgemeine Szenarien implementieren CodeCommit.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`Get-CCBranch`

Das folgende Codebeispiel zeigt, wie Sie es verwenden`Get-CCBranch`.

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Informationen über den angegebenen Branch für das angegebene Repository abgerufen.

```
Get-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch
```

Ausgabe:

```
BranchName                          CommitId
----------                          --------
MyNewBranch                         7763222d...561fc9c9
```

- Einzelheiten zur API finden Sie unter GetBranch AWS Tools for PowerShellCmdlet-Referenz.

`Get-CCBranchList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CCBranchList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste von Branch-Namen für das angegebene Repository abgerufen.

```
Get-CCBranchList -RepositoryName MyDemoRepo
```

Ausgabe:

```
master
MyNewBranch
```

- Einzelheiten zur API finden Sie unter ListBranches AWS Tools for PowerShellCmdlet-Referenz.

`Get-CCRepository`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CCRepository`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Informationen für das angegebene Repository abgerufen.

```
Get-CCRepository -RepositoryName MyDemoRepo
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter GetRepository AWS Tools for PowerShellCmdlet-Referenz.

`Get-CCRepositoryBatch`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CCRepositoryBatch`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird bestätigt, welche der angegebenen Repositorys gefunden wurden und welche nicht.

```
Get-CCRepositoryBatch -RepositoryName MyDemoRepo, MyNewRepo, AMissingRepo
```

Ausgabe:

```
Repositories                            RepositoriesNotFound
------------                            --------------------
{MyDemoRepo, MyNewRepo}                 {AMissingRepo}
```

- Einzelheiten zur API finden Sie unter BatchGetRepositories AWS Tools for PowerShellCmdlet-Referenz.

`Get-CCRepositoryList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CCRepositoryList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden alle Repositorys in aufsteigender Reihenfolge nach dem Repository-Namen aufgelistet.

```
Get-CCRepositoryList -Order Ascending -SortBy RepositoryName
```

Ausgabe:

```
RepositoryId                             RepositoryName
------------                             --------------
c7d0d2b0-ce40-4303-b4c3-38529EXAMPLE     MyDemoRepo
05f30c66-e3e3-4f91-a0cd-1c84aEXAMPLE     MyNewRepo
```

- Einzelheiten zur API finden Sie unter ListRepositories AWS Tools for PowerShellCmdlet-Referenz.

`New-CCBranch`

Das folgende Codebeispiel zeigt die Verwendung. `New-CCBranch`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein neuer Branch mit dem angegebenen Namen für das angegebene Repository und der angegebenen Commit-ID erstellt.

```
New-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch -CommitId
 7763222d...561fc9c9
```

- Einzelheiten zur API finden Sie unter CreateBranch AWS Tools for PowerShellCmdlet-Referenz.

`New-CCRepository`

Das folgende Codebeispiel zeigt die Verwendung. `New-CCRepository`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein neues Repository mit dem angegebenen Namen und der angegebenen Beschreibung erstellt.

```
New-CCRepository -RepositoryName MyDemoRepo -RepositoryDescription "This is a
 repository for demonstration purposes."
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter CreateRepository AWS Tools for PowerShellCmdlet-Referenz.

`Remove-CCRepository`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-CCRepository`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Repository zwangsweise gelöscht. Der Befehl fordert Sie zur Bestätigung auf, bevor Sie fortfahren. Fügen Sie den Parameter -Force hinzu, um das Repository ohne Aufforderung zu löschen.

```
Remove-CCRepository -RepositoryName MyDemoRepo
```

Ausgabe:

```
43ef2443-3372-4b12-9e78-65c27EXAMPLE
```

- Einzelheiten zur API finden Sie unter DeleteRepository AWS Tools for PowerShellCmdlet-Referenz.

`Update-CCDefaultBranch`

Das folgende Codebeispiel zeigt die Verwendung. `Update-CCDefaultBranch`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der Standardzweig für das angegebene Repository in den angegebenen Branch geändert.

```
Update-CCDefaultBranch -RepositoryName MyDemoRepo -DefaultBranchName MyNewBranch
```

- Einzelheiten zur API finden Sie unter UpdateDefaultBranch AWS Tools for PowerShellCmdlet-Referenz.

`Update-CCRepositoryDescription`

Das folgende Codebeispiel zeigt die Verwendung. `Update-CCRepositoryDescription`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Beschreibung für das angegebene Repository geändert.

```
Update-CCRepositoryDescription -RepositoryName MyDemoRepo -RepositoryDescription
 "This is an updated description."
```

- Einzelheiten zur API finden Sie unter UpdateRepositoryDescription AWS Tools for PowerShellCmdlet-Referenz.

`Update-CCRepositoryName`

Das folgende Codebeispiel zeigt die Verwendung. `Update-CCRepositoryName`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der Name des angegebenen Repositorys geändert.

```
Update-CCRepositoryName -NewName MyDemoRepo2 -OldName MyDemoRepo
```

- Einzelheiten zur API finden Sie unter UpdateRepositoryName AWS Tools for PowerShellCmdlet-Referenz.

CodeDeploy Beispiele für die Verwendung von Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS Tools for PowerShell with Aktionen ausführen und allgemeine Szenarien implementieren CodeDeploy.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

Aktionen

`Add-CDOnPremiseInstanceTag`

Das folgende Codebeispiel zeigt, wie Sie es verwendenAdd-CDOnPremiseInstanceTag.

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein On-Premises-Instance-Tag mit dem angegebenen Schlüssel und Wert für die angegebene lokale Instanz hinzugefügt.

```
Add-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" = "Name";
 "Value" = "CodeDeployDemo-OnPrem"}
```

- Einzelheiten zur API finden Sie unter AddTagsToOnPremisesInstances AWS Tools for PowerShellCmdlet-Referenz.

`Get-CDApplication`

Das folgende Codebeispiel zeigt die Verwendung. Get-CDApplication

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Informationen über die angegebene Anwendung abgerufen.

```
Get-CDApplication -ApplicationName CodeDeployDemoApplication
```

Ausgabe:

```
ApplicationId                           ApplicationName             CreateTime
         LinkedToGitHub
-------------                           ---------------             ----------
         --------------
e07fb938-091e-4f2f-8963-4d3e8EXAMPLE    CodeDeployDemoApplication   7/20/2015
 9:49:48 PM    False
```

- Einzelheiten zur API finden Sie unter GetApplication AWS Tools for PowerShellCmdlet-Referenz.

`Get-CDApplicationBatch`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDApplicationBatch`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Informationen zu den angegebenen Anwendungen abgerufen.

```
Get-CDApplicationBatch -ApplicationName CodeDeployDemoApplication,
 CodePipelineDemoApplication
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter BatchGetApplications AWS Tools for PowerShellCmdlet-Referenz.

`Get-CDApplicationList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDApplicationList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste der verfügbaren Anwendungen abgerufen.

```
Get-CDApplicationList
```

Ausgabe:

```
CodeDeployDemoApplication
CodePipelineDemoApplication
```

- Einzelheiten zur API finden Sie unter ListApplications AWS Tools for PowerShellCmdlet-Referenz.

`Get-CDApplicationRevision`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDApplicationRevision`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Informationen über die angegebene Anwendungsrevision abgerufen.

```
$revision = Get-CDApplicationRevision -ApplicationName CodeDeployDemoApplication -
S3Location_Bucket MyBucket -Revision_RevisionType S3 -S3Location_Key 5xd27EX.zip -
S3Location_BundleType zip -S3Location_ETag 4565c1ac97187f190c1a90265EXAMPLE
Write-Output ("Description = " + $revision.RevisionInfo.Description + ",
 RegisterTime = " + $revision.RevisionInfo.RegisterTime)
```

Ausgabe:

```
Description = Application revision registered by Deployment ID: d-CX9CHN3EX,
 RegisterTime = 07/20/2015 23:46:42
```

- Einzelheiten zur API finden Sie unter GetApplicationRevision AWS Tools for PowerShellCmdlet-Referenz.

`Get-CDApplicationRevisionList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDApplicationRevisionList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Informationen zu verfügbaren Versionen für die angegebene Anwendung abgerufen.

```
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)
>>   }
>> }
>>
```

Ausgabe:

```
Type = S3, Bucket = MyBucket, BundleType = zip, ETag =
 4565c1ac97187f190c1a90265EXAMPLE, Key = 5xd27EX.zip
Type = GitHub, CommitId = f48933c3...76405362, Repository = MyGitHubUser/
CodeDeployDemoRepo
```

- Einzelheiten zur API finden Sie unter ListApplicationRevisions AWS Tools for PowerShellCmdlet-Referenz.

`Get-CDDeployment`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDDeployment`

Beispiel 1: In diesem Beispiel werden zusammenfassende Informationen zur angegebenen Bereitstellung abgerufen.

```
Get-CDDeployment -DeploymentId d-QZMRGSTEX
```

Ausgabe:

```
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
```

Beispiel 2: In diesem Beispiel werden Informationen über den Status von Instanzen abgerufen, die an der angegebenen Bereitstellung teilnehmen.

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).DeploymentOverview
```

Ausgabe:

```
Failed     : 0
InProgress : 0
Pending    : 0
Skipped    : 0
Succeeded  : 3
```

Beispiel 3: In diesem Beispiel werden Informationen zur Anwendungsversion für die angegebene Bereitstellung abgerufen.

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).Revision.S3Location
```

Ausgabe:

```
Bucket     : MyBucket
BundleType : zip
ETag       : cfbb81b304ee5e27efc21adaed3EXAMPLE
Key        : clzfqEX
Version    :
```

- Einzelheiten zur API finden Sie unter GetDeployment AWS Tools for PowerShellCmdlet-Referenz.

`Get-CDDeploymentBatch`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDDeploymentBatch`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Informationen zu den angegebenen Bereitstellungen abgerufen.

```
Get-CDDeploymentBatch -DeploymentId d-QZMRGSTEX, d-RR0T5KTEX
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter BatchGetDeployments AWS Tools for PowerShellCmdlet-Referenz.

`Get-CDDeploymentConfig`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDDeploymentConfig`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden zusammenfassende Informationen zur angegebenen Bereitstellungskonfiguration abgerufen.

```
Get-CDDeploymentConfig -DeploymentConfigName ThreeQuartersHealthy
```

Ausgabe:

```
CreateTime              DeploymentConfigId                      DeploymentConfigName
    MinimumHealthyHosts
----------              ------------------                      --------------------
    --------------------
10/3/2014 4:32:30 PM    518a3950-d034-46a1-9d2c-3c949EXAMPLE    ThreeQuartersHealthy
    Amazon.CodeDeploy.Model.MinimumHealthyHosts
```

Beispiel 2: In diesem Beispiel werden Informationen zur Definition der angegebenen Bereitstellungskonfiguration abgerufen.

```
Write-Output ((Get-CDDeploymentConfig -DeploymentConfigName
 ThreeQuartersHealthy).MinimumHealthyHosts)
```

Ausgabe:

```
Type             Value
----             -----
FLEET_PERCENT    75
```

- Einzelheiten zur API finden Sie unter GetDeploymentConfig AWS Tools for PowerShellCmdlet-Referenz.

`Get-CDDeploymentConfigList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDDeploymentConfigList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste der verfügbaren Bereitstellungskonfigurationen abgerufen.

```
Get-CDDeploymentConfigList
```

Ausgabe:

```
ThreeQuartersHealthy
CodeDeployDefault.OneAtATime
CodeDeployDefault.AllAtOnce
CodeDeployDefault.HalfAtATime
```

- Einzelheiten zur API finden Sie unter ListDeploymentConfigs AWS Tools for PowerShellCmdlet-Referenz.

`Get-CDDeploymentGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDDeploymentGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Informationen über die angegebene Bereitstellungsgruppe abgerufen.

```
Get-CDDeploymentGroup -ApplicationName CodeDeployDemoApplication -
DeploymentGroupName CodeDeployDemoFleet
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter GetDeploymentGroup AWS Tools for PowerShellCmdlet-Referenz.

`Get-CDDeploymentGroupList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDDeploymentGroupList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste von Bereitstellungsgruppen für die angegebene Anwendung abgerufen.

```
Get-CDDeploymentGroupList -ApplicationName CodeDeployDemoApplication
```

Ausgabe:

```
ApplicationName             DeploymentGroups
 NextToken
---------------             ----------------
 ---------
CodeDeployDemoApplication    {CodeDeployDemoFleet, CodeDeployProductionFleet}
```

- Einzelheiten zur API finden Sie unter ListDeploymentGroups AWS Tools for PowerShellCmdlet-Referenz.

`Get-CDDeploymentInstance`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDDeploymentInstance`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Informationen über die angegebene Instanz für die angegebene Bereitstellung abgerufen.

```
Get-CDDeploymentInstance -DeploymentId d-QZMRGSTEX -InstanceId i-254e22EX
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter GetDeploymentInstance AWS Tools for PowerShellCmdlet-Referenz.

`Get-CDDeploymentInstanceList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDDeploymentInstanceList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste von Instanz-IDs für die angegebene Bereitstellung abgerufen.

```
Get-CDDeploymentInstanceList -DeploymentId d-QZMRGSTEX
```

Ausgabe:

```
i-254e22EX
i-274e22EX
i-3b4e22EX
```

- Einzelheiten zur API finden Sie unter ListDeploymentInstances AWS Tools for PowerShellCmdlet-Referenz.

`Get-CDDeploymentList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDDeploymentList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste von Bereitstellungs-IDs für die angegebene Anwendung und Bereitstellungsgruppe abgerufen.

```
Get-CDDeploymentList -ApplicationName CodeDeployDemoApplication -DeploymentGroupName
 CodeDeployDemoFleet
```

Ausgabe:

```
d-QZMRGSTEX
d-RR0T5KTEX
```

- Einzelheiten zur API finden Sie unter ListDeployments AWS Tools for PowerShellCmdlet-Referenz.

`Get-CDOnPremiseInstance`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDOnPremiseInstance`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Informationen über die angegebene lokale Instanz abgerufen.

```
Get-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

Ausgabe:

```
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}
```

- Einzelheiten zur API finden Sie unter GetOnPremisesInstance AWS Tools for PowerShellCmdlet-Referenz.

`Get-CDOnPremiseInstanceBatch`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDOnPremiseInstanceBatch`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Informationen zu den angegebenen lokalen Instanzen abgerufen.

```
Get-CDOnPremiseInstanceBatch -InstanceName AssetTag12010298EX, AssetTag12010298EX-2
```

Ausgabe:

```
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}
```

- Einzelheiten zur API finden Sie unter BatchGetOnPremisesInstances AWS Tools for PowerShellCmdlet-Referenz.

`Get-CDOnPremiseInstanceList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDOnPremiseInstanceList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste der verfügbaren lokalen Instanznamen abgerufen.

```
Get-CDOnPremiseInstanceList
```

Ausgabe:

```
AssetTag12010298EX
AssetTag12010298EX-2
```

- Einzelheiten zur API finden Sie unter ListOnPremisesInstances AWS Tools for PowerShellCmdlet-Referenz.

`New-CDApplication`

Das folgende Codebeispiel zeigt die Verwendung. `New-CDApplication`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine neue Anwendung mit dem angegebenen Namen erstellt.

```
New-CDApplication -ApplicationName MyNewApplication
```

Ausgabe:

```
f19e4b61-2231-4328-b0fd-e57f5EXAMPLE
```

- Einzelheiten zur API finden Sie unter CreateApplication AWS Tools for PowerShellCmdlet-Referenz.

`New-CDDeployment`

Das folgende Codebeispiel zeigt die Verwendung. `New-CDDeployment`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine neue Bereitstellung für die angegebene Anwendung und Bereitstellungsgruppe mit der angegebenen Bereitstellungskonfiguration und Anwendungsversion erstellt.

```
New-CDDeployment -ApplicationName MyNewApplication -S3Location_Bucket MyBucket
 -S3Location_BundleType zip -DeploymentConfigName CodeDeployDefault.OneAtATime -
DeploymentGroupName MyNewDeploymentGroup -IgnoreApplicationStopFailures $True -
S3Location_Key aws-codedeploy_linux-master.zip -RevisionType S3
```

Ausgabe:

```
d-ZHROG7UEX
```

Beispiel 2: Dieses Beispiel zeigt, wie Gruppen von EC2-Instance-Tags angegeben werden, anhand derer eine Instance identifiziert werden muss, damit sie in die Ersatzumgebung für eine blaue/grüne Bereitstellung aufgenommen werden kann.

```
New-CDDeployment -ApplicationName MyNewApplication -S3Location_Bucket MyBucket
 -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=
```

Ausgabe:

```
d-ZHROG7UEX
```

- Einzelheiten zur API finden Sie unter CreateDeployment AWS Tools for PowerShellCmdlet-Referenz.

`New-CDDeploymentConfig`

Das folgende Codebeispiel zeigt die Verwendung. `New-CDDeploymentConfig`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine neue Bereitstellungskonfiguration mit dem angegebenen Namen und Verhalten erstellt.

```
New-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts -
MinimumHealthyHosts_Type HOST_COUNT -MinimumHealthyHosts_Value 2
```

Ausgabe:

```
0f3e8187-44ef-42da-aeed-b6823EXAMPLE
```

- Einzelheiten zur API finden Sie unter CreateDeploymentConfig AWS Tools for PowerShellCmdlet-Referenz.

`New-CDDeploymentGroup`

Das folgende Codebeispiel zeigt die Verwendung. `New-CDDeploymentGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Bereitstellungsgruppe mit dem angegebenen Namen, der Auto Scaling Scaling-Gruppe, der Bereitstellungskonfiguration, dem Tag und der Servicerolle für die angegebene Anwendung erstellt.

```
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
```

Ausgabe:

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

Beispiel 2: Dieses Beispiel zeigt, wie Gruppen von EC2-Instance-Tags angegeben werden, anhand derer eine Instance identifiziert werden muss, damit sie in die Ersatzumgebung für eine blaue/grüne Bereitstellung aufgenommen werden kann.

```
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=
```

Ausgabe:

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

- Einzelheiten zur API finden Sie unter CreateDeploymentGroup AWS Tools for PowerShellCmdlet-Referenz.

`Register-CDApplicationRevision`

Das folgende Codebeispiel zeigt die Verwendung. `Register-CDApplicationRevision`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Anwendungsrevision mit dem angegebenen Amazon S3 S3-Standort für die angegebene Anwendung registriert.

```
Register-CDApplicationRevision -ApplicationName MyNewApplication -S3Location_Bucket
 MyBucket -S3Location_BundleType zip -S3Location_Key aws-codedeploy_linux-master.zip
 -Revision_RevisionType S3
```

- Einzelheiten zur API finden Sie unter RegisterApplicationRevision AWS Tools for PowerShellCmdlet-Referenz.

`Register-CDOnPremiseInstance`

Das folgende Codebeispiel zeigt die Verwendung. `Register-CDOnPremiseInstance`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine lokale Instanz mit dem angegebenen Namen und dem angegebenen IAM-Benutzer registriert.

```
Register-CDOnPremiseInstance -IamUserArn arn:aws:iam::80398EXAMPLE:user/
CodeDeployDemoUser -InstanceName AssetTag12010298EX
```

- Einzelheiten zur API finden Sie unter RegisterOnPremisesInstance AWS Tools for PowerShellCmdlet-Referenz.

`Remove-CDApplication`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-CDApplication`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Anwendung mit dem angegebenen Namen gelöscht. Der Befehl fordert Sie zur Bestätigung auf, bevor Sie fortfahren. Fügen Sie den Parameter -Force hinzu, um die Anwendung ohne Aufforderung zu löschen.

```
Remove-CDApplication -ApplicationName MyNewApplication
```

- Einzelheiten zur API finden Sie unter DeleteApplication AWS Tools for PowerShellCmdlet-Referenz.

`Remove-CDDeploymentConfig`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-CDDeploymentConfig`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Bereitstellungskonfiguration mit dem angegebenen Namen gelöscht. Der Befehl fordert Sie zur Bestätigung auf, bevor Sie fortfahren. Fügen Sie den Parameter -Force hinzu, um die Bereitstellungskonfiguration ohne Aufforderung zu löschen.

```
Remove-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts
```

- Einzelheiten zur API finden Sie unter DeleteDeploymentConfig AWS Tools for PowerShellCmdlet-Referenz.

`Remove-CDDeploymentGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-CDDeploymentGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Bereitstellungsgruppe mit dem angegebenen Namen für die angegebene Anwendung gelöscht. Der Befehl fordert Sie zur Bestätigung auf, bevor Sie fortfahren. Fügen Sie den Parameter -Force hinzu, um die Bereitstellungsgruppe ohne Aufforderung zu löschen.

```
Remove-CDDeploymentGroup -ApplicationName MyNewApplication -DeploymentGroupName
 MyNewDeploymentGroup
```

- Einzelheiten zur API finden Sie unter DeleteDeploymentGroup AWS Tools for PowerShellCmdlet-Referenz.

`Remove-CDOnPremiseInstanceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-CDOnPremiseInstanceTag`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Tag für die lokale Instanz mit dem angegebenen Namen gelöscht. Der Befehl fordert Sie zur Bestätigung auf, bevor Sie fortfahren. Fügen Sie den Parameter -Force hinzu, um das Tag ohne Aufforderung zu löschen.

```
Remove-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" =
 "Name"; "Value" = "CodeDeployDemo-OnPrem"}
```

- Einzelheiten zur API finden Sie unter RemoveTagsFromOnPremisesInstances AWS Tools for PowerShellCmdlet-Referenz.

`Stop-CDDeployment`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-CDDeployment`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird versucht, die Bereitstellung mit der angegebenen Bereitstellungs-ID zu beenden.

```
Stop-CDDeployment -DeploymentId d-LJQNREYEX
```

Ausgabe:

```
Status     StatusMessage
------     -------------
Pending    Stopping Pending. Stopping to schedule commands in the deployment
 instances
```

- Einzelheiten zur API finden Sie unter StopDeployment AWS Tools for PowerShellCmdlet-Referenz.

`Unregister-CDOnPremiseInstance`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-CDOnPremiseInstance`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Registrierung der lokalen Instanz mit dem angegebenen Namen aufgehoben.

```
Unregister-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

- Einzelheiten zur API finden Sie unter DeregisterOnPremisesInstanceCmdlet-Referenz.AWS Tools for PowerShell

`Update-CDApplication`

Das folgende Codebeispiel zeigt die Verwendung. `Update-CDApplication`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der Name der angegebenen Anwendung geändert.

```
Update-CDApplication -ApplicationName MyNewApplication -NewApplicationName
 MyNewApplication-2
```

- Einzelheiten zur API finden Sie unter UpdateApplication AWS Tools for PowerShellCmdlet-Referenz.

`Update-CDDeploymentGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Update-CDDeploymentGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der Name der angegebenen Bereitstellungsgruppe für die angegebene Anwendung geändert.

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication -
CurrentDeploymentGroupName MyNewDeploymentGroup -NewDeploymentGroupName
 MyNewDeploymentGroup-2
```

Beispiel 2: Dieses Beispiel zeigt, wie Gruppen von EC2-Instance-Tags angegeben werden, anhand derer eine Instance identifiziert werden muss, damit sie in die Ersatzumgebung für eine blaue/grüne Bereitstellung aufgenommen werden kann.

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication -
CurrentDeploymentGroupName MyNewDeploymentGroup -NewDeploymentGroupName
 MyNewDeploymentGroup-2 -Ec2TagSetList
 @(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{Key=
```

- Einzelheiten zur API finden Sie unter UpdateDeploymentGroup AWS Tools for PowerShellCmdlet-Referenz.

CodePipeline Beispiele für die Verwendung von Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS Tools for PowerShell with Aktionen ausführen und allgemeine Szenarien implementieren CodePipeline.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`Confirm-CPJob`

Das folgende Codebeispiel zeigt, wie Sie es verwenden`Confirm-CPJob`.

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der Status des angegebenen Jobs abgerufen.

```
Confirm-CPJob -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE -Nonce 3
```

Ausgabe:

```
Value
-----
InProgress
```

- Einzelheiten zur API finden Sie unter AcknowledgeJob AWS Tools for PowerShellCmdlet-Referenz.

`Disable-CPStageTransition`

Das folgende Codebeispiel zeigt die Verwendung. `Disable-CPStageTransition`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der eingehende Übergang für die angegebene Phase in der angegebenen Pipeline deaktiviert.

```
Disable-CPStageTransition -PipelineName CodePipelineDemo -Reason "Disabling
 temporarily." -StageName Beta -TransitionType Inbound
```

- Einzelheiten zur API finden Sie unter DisableStageTransition AWS Tools for PowerShellCmdlet-Referenz.

`Enable-CPStageTransition`

Das folgende Codebeispiel zeigt die Verwendung. `Enable-CPStageTransition`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der eingehende Übergang für die angegebene Phase in der angegebenen Pipeline aktiviert.

```
Enable-CPStageTransition -PipelineName CodePipelineDemo -StageName Beta -
TransitionType Inbound
```

- Einzelheiten zur API finden Sie unter EnableStageTransition AWS Tools for PowerShellCmdlet-Referenz.

`Get-CPActionType`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CPActionType`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Informationen über alle verfügbaren Aktionen für den angegebenen Besitzer abgerufen.

```
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)
}
```

Ausgabe:

```
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}
```

- Einzelheiten zur API finden Sie unter ListActionTypes AWS Tools for PowerShellCmdlet-Referenz.

`Get-CPActionableJobList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CPActionableJobList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Informationen über alle Aufträge abgerufen, für die eine Aktion in der angegebenen Kategorie, dem Besitzer, dem Anbieter, der Version und den Abfrageparametern ausgeführt werden kann.

```
Get-CPActionableJobList -ActionTypeId_Category Build -ActionTypeId_Owner Custom
 -ActionTypeId_Provider MyCustomProviderName -ActionTypeId_Version 1 -QueryParam
 @{"ProjectName" = "MyProjectName"}
```

Ausgabe:

```
AccountId       Data                                    Id
         Nonce
---------       ----                                    --
         -----
80398EXAMPLE    Amazon.CodePipeline.Model.JobData       0de392f5-712d-4f41-ace3-
f57a0EXAMPLE    3
```

- Einzelheiten zur API finden Sie unter PollForJobs AWS Tools for PowerShellCmdlet-Referenz.

`Get-CPJobDetail`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CPJobDetail`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden allgemeine Informationen über den angegebenen Job abgerufen.

```
Get-CPJobDetail -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE
```

Ausgabe:

```
AccountId       Data                                    Id
---------       ----                                    --
80398EXAMPLE    Amazon.CodePipeline.Model.JobData
 f570dc12-5ef3-44bc-945a-6e133EXAMPLE
```

Beispiel 2: In diesem Beispiel werden detaillierte Informationen über den angegebenen Job abgerufen.

```
$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)
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter GetJobDetails AWS Tools for PowerShellCmdlet-Referenz.

`Get-CPPipeline`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CPPipeline`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden allgemeine Informationen über die angegebene Pipeline abgerufen.

```
Get-CPPipeline -Name CodePipelineDemo -Version 1
```

Ausgabe:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {Source, Build, Beta, TestStage}
Version       : 1
```

Beispiel 2: In diesem Beispiel werden detaillierte Informationen über die angegebene Pipeline abgerufen.

```
$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)
  }
}
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter GetPipeline AWS Tools for PowerShellCmdlet-Referenz.

`Get-CPPipelineList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CPPipelineList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste der verfügbaren Pipelines abgerufen.

```
Get-CPPipelineList
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter ListPipelines AWS Tools for PowerShellCmdlet-Referenz.

`Get-CPPipelineState`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CPPipelineState`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden allgemeine Informationen zu den Phasen der angegebenen Pipeline abgerufen.

```
Get-CPPipelineState -Name CodePipelineDemo
```

Ausgabe:

```
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
```

Beispiel 2: In diesem Beispiel werden detaillierte Informationen zum Status der angegebenen Pipeline abgerufen.

```
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)
  }
}
```

Ausgabe:

```
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 =
```

- Einzelheiten zur API finden Sie unter GetPipelineState AWS Tools for PowerShellCmdlet-Referenz.

`New-CPCustomActionType`

Das folgende Codebeispiel zeigt die Verwendung. `New-CPCustomActionType`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine neue benutzerdefinierte Aktion mit den angegebenen Eigenschaften erstellt.

```
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
```

Ausgabe:

```
ActionConfigurationProperties : {ProjectName}
Id                            : Amazon.CodePipeline.Model.ActionTypeId
InputArtifactDetails          : Amazon.CodePipeline.Model.ArtifactDetails
OutputArtifactDetails         : Amazon.CodePipeline.Model.ArtifactDetails
Settings                      : Amazon.CodePipeline.Model.ActionTypeSettings
```

- Einzelheiten zur API finden Sie unter CreateCustomActionType AWS Tools for PowerShellCmdlet-Referenz.

`New-CPPipeline`

Das folgende Codebeispiel zeigt die Verwendung. `New-CPPipeline`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine neue Pipeline mit den angegebenen Einstellungen erstellt.

```
$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", "MyBucketName")
$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" = "MyBucketName"; "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
```

Ausgabe:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
```

```
Stages        : {Source, Beta}
Version       : 1
```

- Einzelheiten zur API finden Sie unter CreatePipeline AWS Tools for PowerShellCmdlet-Referenz.

`Remove-CPCustomActionType`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-CPCustomActionType`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene benutzerdefinierte Aktion gelöscht. Der Befehl fordert Sie zur Bestätigung auf, bevor Sie fortfahren. Fügen Sie den Parameter -Force hinzu, um die benutzerdefinierte Aktion ohne Aufforderung zu löschen.

```
Remove-CPCustomActionType -Category Build -Provider MyBuildProviderName -Version 1
```

- Einzelheiten zur API finden Sie unter DeleteCustomActionType AWS Tools for PowerShellCmdlet-Referenz.

`Remove-CPPipeline`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-CPPipeline`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Pipeline gelöscht. Der Befehl fordert Sie zur Bestätigung auf, bevor Sie fortfahren. Fügen Sie den Parameter -Force hinzu, um die Pipeline ohne Aufforderung zu löschen.

```
Remove-CPPipeline -Name CodePipelineDemo
```

- Einzelheiten zur API finden Sie unter DeletePipeline AWS Tools for PowerShellCmdlet-Referenz.

`Start-CPPipelineExecution`

Das folgende Codebeispiel zeigt die Verwendung. `Start-CPPipelineExecution`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird mit der Ausführung der angegebenen Pipeline begonnen.

```
Start-CPPipelineExecution -Name CodePipelineDemo
```

- Einzelheiten zur API finden Sie unter StartPipelineExecution AWS Tools for PowerShellCmdlet-Referenz.

`Update-CPPipeline`

Das folgende Codebeispiel zeigt die Verwendung. `Update-CPPipeline`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene bestehende Pipeline mit den angegebenen Einstellungen aktualisiert.

```
$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", "MyBucketName")
$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" = "MyBucketName"; "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
```

Ausgabe:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {InputFiles, TestDeployment}
Version       : 2
```

- Einzelheiten zur API finden Sie unter UpdatePipeline AWS Tools for PowerShellCmdlet-Referenz.

Beispiele für Amazon Cognito Identity mit Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS Tools for PowerShell mit Amazon Cognito Identity Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zur Einrichtung und Ausführung des Codes im Kontext finden.

Themen

Aktionen

`Get-CGIIdentityPool`

Das folgende Codebeispiel zeigt, wie Sie es verwenden`Get-CGIIdentityPool`.

Tools für PowerShell

Beispiel 1: Ruft Informationen über einen bestimmten Identitätspool anhand seiner ID ab.

```
Get-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeIdentityPool AWS Tools for PowerShellCmdlet-Referenz.

`Get-CGIIdentityPoolList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CGIIdentityPoolList`

Tools für PowerShell

Beispiel 1: Ruft eine Liste vorhandener Identitätspools ab.

```
Get-CGIIdentityPoolList
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter ListIdentityPools AWS Tools for PowerShellCmdlet-Referenz.

`Get-CGIIdentityPoolRole`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CGIIdentityPoolRole`

Tools für PowerShell

Beispiel 1: Ruft die Informationen zu Rollen für einen bestimmten Identitätspool ab.

```
Get-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter GetIdentityPoolRoles AWS Tools for PowerShellCmdlet-Referenz.

`New-CGIIdentityPool`

Das folgende Codebeispiel zeigt die Verwendung. `New-CGIIdentityPool`

Tools für PowerShell

Beispiel 1: Erstellt einen neuen Identitätspool, der nicht authentifizierte Identitäten zulässt.

```
New-CGIIdentityPool -AllowUnauthenticatedIdentities $true -IdentityPoolName
 CommonTests13
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter CreateIdentityPoolCmdlet-Referenz.AWS Tools for PowerShell

`Remove-CGIIdentityPool`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-CGIIdentityPool`

Tools für PowerShell

Beispiel 1: Löscht einen bestimmten Identitätspool.

```
Remove-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

- Einzelheiten zur API finden Sie unter DeleteIdentityPool AWS Tools for PowerShellCmdlet-Referenz.

`Set-CGIIdentityPoolRole`

Das folgende Codebeispiel zeigt die Verwendung. `Set-CGIIdentityPoolRole`

Tools für PowerShell

Beispiel 1: Konfiguriert den spezifischen Identity Pool so, dass er eine nicht authentifizierte IAM-Rolle hat.

```
Set-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -Role @{ "unauthenticated" = "arn:aws:iam::123456789012:role/
CommonTests1Role" }
```

- Einzelheiten zur API finden Sie unter SetIdentityPoolRolesCmdlet-Referenz.AWS Tools for PowerShell

`Update-CGIIdentityPool`

Das folgende Codebeispiel zeigt die Verwendung. `Update-CGIIdentityPool`

Tools für PowerShell

Beispiel 1: Aktualisiert einige Eigenschaften des Identitätspools, in diesem Fall den Namen des Identitätspools.

```
Update-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -IdentityPoolName NewPoolName
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter UpdateIdentityPool AWS Tools for PowerShellCmdlet-Referenz.

AWS Config Beispiele für die Verwendung von Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS Tools for PowerShell with Aktionen ausführen und allgemeine Szenarien implementieren AWS Config.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`Add-CFGResourceTag`

Das folgende Codebeispiel zeigt, wie Sie es verwenden`Add-CFGResourceTag`.

Tools für PowerShell

Beispiel 1: Dieses Beispiel ordnet das angegebene Tag dem Ressourcen-ARN zu, in diesem Fall config-rule/config-rule-16iyn0.

```
Add-CFGResourceTag -ResourceArn arn:aws:config:eu-west-1:123456789012:config-rule/
config-rule-16iyn0 -Tag @{Key="Release";Value="Beta"}
```

- Einzelheiten TagResourcezur AWS Tools for PowerShell API finden Sie unter Cmdlet-Referenz.

`Get-CFGAggregateComplianceByConfigRuleList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGAggregateComplianceByConfigRuleList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Details aus der ConfigurationAggregator „Kaju“ - Filterung für die angegebene Konfigurationsregel abgerufen und die „Konformität“ der Regel erweitert/zurückgegeben.

```
Get-CFGAggregateComplianceByConfigRuleList -ConfigurationAggregatorName kaju
 -Filters_ConfigRuleName ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK | Select-Object -
ExpandProperty Compliance
```

Ausgabe:

```
ComplianceContributorCount                          ComplianceType
--------------------------                          --------------
Amazon.ConfigService.Model.ComplianceContributorCount NON_COMPLIANT
```

Beispiel 2: In diesem Beispiel werden Details aus der angegebenen Datei abgerufen ConfigurationAggregator, nach dem angegebenen Konto für alle im Aggregator abgedeckten Regionen gefiltert und anschließend die Konformität für alle Regeln zurückgegeben.

```
Get-CFGAggregateComplianceByConfigRuleList -ConfigurationAggregatorName
 kaju -Filters_AccountId 123456789012 | Select-Object ConfigRuleName,
 @{N="Compliance";E={$_.Compliance.ComplianceType}}
```

Ausgabe:

```
ConfigRuleName                    Compliance
--------------                    ----------
ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK NON_COMPLIANT
ec2-instance-no-public-ip         NON_COMPLIANT
desired-instance-type             NON_COMPLIANT
```

- Einzelheiten zur API finden Sie unter DescribeAggregateComplianceByConfigRulesCmdlet-Referenz.AWS Tools for PowerShell

`Get-CFGAggregateComplianceDetailsByConfigRule`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGAggregateComplianceDetailsByConfigRule`

Tools für PowerShell

Beispiel 1: Dieses Beispiel gibt die Auswertungsergebnisse zurück, indem die Ausgabe mit der Ressourcen-ID und dem Ressourcentyp für die AWS Konfigurationsregel 'desired-instance-type' ausgewählt wird, die sich für das angegebene Konto, den Aggregator, die Region und die Konfigurationsregel im Status 'COMPLIANT' befinden

```
Get-CFGAggregateComplianceDetailsByConfigRule -AccountId 123456789012 -
AwsRegion eu-west-1 -ComplianceType COMPLIANT -ConfigRuleName desired-
instance-type -ConfigurationAggregatorName raju | Select-Object -
ExpandProperty EvaluationResultIdentifier | Select-Object -ExpandProperty
 EvaluationResultQualifier
```

Ausgabe:

```
ConfigRuleName        ResourceId          ResourceType
--------------        ----------          ------------
desired-instance-type i-0f1bf2f34c5678d12 AWS::EC2::Instance
desired-instance-type i-0fd12dd3456789123 AWS::EC2::Instance
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz. GetAggregateComplianceDetailsByConfigRuleAWS Tools for PowerShell

`Get-CFGAggregateConfigRuleComplianceSummary`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGAggregateConfigRuleComplianceSummary`

Tools für PowerShell

Beispiel 1: Dieses Beispiel gibt die Anzahl der nicht konformen Regeln für den angegebenen Aggregator zurück.

```
(Get-CFGAggregateConfigRuleComplianceSummary -ConfigurationAggregatorName
 raju).AggregateComplianceCounts.ComplianceSummary.NonCompliantResourceCount
```

Ausgabe:

```
CapExceeded CappedCount
----------- -----------
False       5
```

- Einzelheiten zur API finden Sie unter GetAggregateConfigRuleComplianceSummary AWS Tools for PowerShellCmdlet-Referenz.

`Get-CFGAggregateDiscoveredResourceCount`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGAggregateDiscoveredResourceCount`

Tools für PowerShell

Beispiel 1: Dieses Beispiel gibt die Ressourcenanzahl für den angegebenen Aggregator zurück, gefiltert nach der Region us-east-1.

```
Get-CFGAggregateDiscoveredResourceCount -ConfigurationAggregatorName Master -
Filters_Region us-east-1
```

Ausgabe:

```
GroupByKey GroupedResourceCounts NextToken TotalDiscoveredResources
---------- -------------------- --------- -----------------------
           {}                              455
```

Beispiel 2: Dieses Beispiel gibt die Ressourcenanzahl gruppiert nach RESOURCE_TYPE für die gefilterte Region für den angegebenen Aggregator zurück.

```
Get-CFGAggregateDiscoveredResourceCount -ConfigurationAggregatorName Master -
Filters_Region us-east-1 -GroupByKey RESOURCE_TYPE |
   Select-Object -ExpandProperty GroupedResourceCounts
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter GetAggregateDiscoveredResourceCountsCmdlet-Referenz.AWS Tools for PowerShell

`Get-CFGAggregateDiscoveredResourceList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGAggregateDiscoveredResourceList`

Tools für PowerShell

Beispiel 1: Dieses Beispiel gibt die Ressourcen-IDs für den angegebenen Ressourcentyp zurück, aggregiert im Aggregator „Irland“. Die Liste der Ressourcentypen finden Sie unter https://docs.aws.amazon.com/sdkfornet/v3/apidocs/index.html?page = ConfigService / T .html&tocid=Amazon_ _ConfigServiceResourceType. ConfigService ResourceType

```
Get-CFGAggregateDiscoveredResourceList -ConfigurationAggregatorName Ireland -
ResourceType ([Amazon.ConfigService.ResourceType]::AWSAutoScalingAutoScalingGroup)
```

Ausgabe:

```
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
```

Beispiel 2: Dieses Beispiel gibt den Ressourcentyp mit **`AwsEC2SecurityGroup`** dem Namen 'default' für den angegebenen Aggregator zurück, gefiltert mit der Region us-east-1.

```
Get-CFGAggregateDiscoveredResourceList -ConfigurationAggregatorName raju -
ResourceType ([Amazon.ConfigService.ResourceType]::AWSEC2SecurityGroup) -
Filters_Region us-east-1 -Filters_ResourceName default
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter ListAggregateDiscoveredResourcesCmdlet-Referenz.AWS Tools for PowerShell

`Get-CFGAggregateResourceConfig`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGAggregateResourceConfig`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Konfigurationselement für die angegebene Ressource aggregiert zurückgegeben und die Konfiguration erweitert.

```
(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
```

Ausgabe:

```
{"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"}
```

- Einzelheiten zur API finden Sie unter GetAggregateResourceconfig-service in der AWS Tools for PowerShell Cmdlet-Referenz.

`Get-CFGAggregateResourceConfigBatch`

Das folgende Codebeispiel zeigt, wie Sie es verwenden. `Get-CFGAggregateResourceConfigBatch`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das aktuelle Konfigurationselement für die Ressource (identifiziert) abgerufen, die im angegebenen Aggregator vorhanden ist.

```
$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
```

Ausgabe:

```
BaseConfigurationItems UnprocessedResourceIdentifiers
---------------------- ------------------------------
{}                     {arn:aws:ec2:eu-west-1:123456789012:instance/
i-012e3cb4df567e8aa}
```

- Einzelheiten zur API finden Sie unter BatchGetAggregateResourceconfig-service in AWS Tools for PowerShell der Cmdlet-Referenz.

`Get-CFGAggregationAuthorizationList`

Das folgende Codebeispiel zeigt, wie Sie es verwenden. `Get-CFGAggregationAuthorizationList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Autorisierungen abgerufen, die Aggregatoren erteilt wurden.

```
Get-CFGAggregationAuthorizationList
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeAggregationAuthorizationsCmdlet-Referenz.AWS Tools for PowerShell

`Get-CFGComplianceByConfigRule`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGComplianceByConfigRule`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Konformitätsdetails für die Regel abgerufen ebs-optimized-instance, für die es keine aktuellen Bewertungsergebnisse für die Regel gibt. Daher wird INSUFFICIENT_DATA zurückgegeben

```
(Get-CFGComplianceByConfigRule -ConfigRuleName ebs-optimized-instance).Compliance
```

Ausgabe:

```
ComplianceContributorCount ComplianceType
-------------------------- --------------
                           INSUFFICIENT_DATA
```

Beispiel 2: Dieses Beispiel gibt die Anzahl der nicht konformen Ressourcen für die Regel ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK zurück.

```
(Get-CFGComplianceByConfigRule -ConfigRuleName ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK -
ComplianceType NON_COMPLIANT).Compliance.ComplianceContributorCount
```

Ausgabe:

```
CapExceeded CappedCount
----------- -----------
False       2
```

- Einzelheiten AWS Tools for PowerShell zur DescribeComplianceByConfigRuleAPI finden Sie unter Cmdlet-Referenz.

`Get-CFGComplianceByResource`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGComplianceByResource`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der **`AWS::SSM::ManagedInstanceInventory`** Ressourcentyp auf den Konformitätstyp „COMPLIANT“ geprüft.

```
Get-CFGComplianceByResource -ComplianceType COMPLIANT -ResourceType
 AWS::SSM::ManagedInstanceInventory
```

Ausgabe:

```
Compliance                              ResourceId          ResourceType
----------                              ----------          ------------
Amazon.ConfigService.Model.Compliance i-0123bcf4b567890e3
 AWS::SSM::ManagedInstanceInventory
Amazon.ConfigService.Model.Compliance i-0a1234f6f5d6b78f7
 AWS::SSM::ManagedInstanceInventory
```

- Einzelheiten zur API finden Sie unter DescribeComplianceByResource AWS Tools for PowerShellCmdlet-Referenz.

`Get-CFGComplianceDetailsByConfigRule`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGComplianceDetailsByConfigRule`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Auswertungsergebnisse für die Regel abgerufen access-keys-rotated und die Ausgabe nach Konformitätstyp gruppiert zurückgegeben

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-keys-rotated | Group-
Object ComplianceType
```

Ausgabe:

```
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...
```

Beispiel 2: In diesem Beispiel werden Konformitätsdetails für die Regel access-keys-rotated für COMPLIANT-Ressourcen abgefragt.

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-
keys-rotated -ComplianceType COMPLIANT | ForEach-Object
 {$_.EvaluationResultIdentifier.EvaluationResultQualifier}
```

Ausgabe:

```
ConfigRuleName      ResourceId            ResourceType
--------------      ----------            ------------
access-keys-rotated BCAB1CDJ2LITAPVEW3JAH AWS::IAM::User
access-keys-rotated BCAB1CDJ2LITL3EHREM4Q AWS::IAM::User
```

- Einzelheiten zur API finden Sie unter GetComplianceDetailsByConfigRule AWS Tools for PowerShellCmdlet-Referenz.

`Get-CFGComplianceDetailsByResource`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGComplianceDetailsByResource`

Tools für PowerShell

Beispiel 1: Die Ergebnisse dieser Beispielauswertung für die angegebene Ressource.

```
Get-CFGComplianceDetailsByResource -ResourceId ABCD5STJ4EFGHIVEW6JAH -ResourceType
 'AWS::IAM::User'
```

Ausgabe:

```
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                :
```

- Einzelheiten zur API finden Sie unter GetComplianceDetailsByResource AWS Tools for PowerShellCmdlet-Referenz.

`Get-CFGComplianceSummaryByConfigRule`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGComplianceSummaryByConfigRule`

Tools für PowerShell

Beispiel 1: Dieses Beispiel gibt die Anzahl der Config-Regeln zurück, die nicht konform sind.

```
Get-CFGComplianceSummaryByConfigRule -Select
 ComplianceSummary.NonCompliantResourceCount
```

Ausgabe:

```
CapExceeded CappedCount
----------- -----------
False       9
```

- Einzelheiten zur API finden Sie unter GetComplianceSummaryByConfigRule AWS Tools for PowerShellCmdlet-Referenz.

`Get-CFGComplianceSummaryByResourceType`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGComplianceSummaryByResourceType`

Tools für PowerShell

Beispiel 1: Dieses Beispiel gibt die Anzahl der Ressourcen zurück, die konform oder nicht konform sind, und konvertiert die Ausgabe in 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
  }
}
```

- Einzelheiten zur API finden Sie unter GetComplianceSummaryByResourceType AWS Tools for PowerShellCmdlet-Referenz.

`Get-CFGConfigRule`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGConfigRule`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Konfigurationsregeln für das Konto mit ausgewählten Eigenschaften aufgeführt.

```
Get-CFGConfigRule | Select-Object ConfigRuleName, ConfigRuleId, ConfigRuleArn,
 ConfigRuleState
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeConfigRules AWS Tools for PowerShellCmdlet-Referenz.

`Get-CFGConfigRuleEvaluationStatus`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGConfigRuleEvaluationStatus`

Tools für PowerShell

Beispiel 1: Dieses Beispiel gibt die Statusinformationen für die angegebenen Konfigurationsregeln zurück.

```
Get-CFGConfigRuleEvaluationStatus -ConfigRuleName root-account-mfa-enabled, vpc-
flow-logs-enabled
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeConfigRuleEvaluationStatus AWS Tools for PowerShellCmdlet-Referenz.

`Get-CFGConfigurationAggregatorList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGConfigurationAggregatorList`

Tools für PowerShell

Beispiel 1: Dieses Beispiel gibt alle Aggregatoren für die Region/das Konto zurück.

```
Get-CFGConfigurationAggregatorList
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeConfigurationAggregatorsCmdlet-Referenz.AWS Tools for PowerShell

`Get-CFGConfigurationAggregatorSourcesStatus`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGConfigurationAggregatorSourcesStatus`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden angeforderte Felder für die Quellen im angegebenen Aggregator angezeigt.

```
Get-CFGConfigurationAggregatorSourcesStatus -ConfigurationAggregatorName raju |
 select SourceType, LastUpdateStatus, LastUpdateTime, SourceId
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeConfigurationAggregatorSourcesStatus AWS Tools for PowerShellCmdlet-Referenz.

`Get-CFGConfigurationRecorder`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGConfigurationRecorder`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Details von Konfigurationsrekordern zurückgegeben.

```
Get-CFGConfigurationRecorder | Format-List
```

Ausgabe:

```
Name           : default
RecordingGroup : Amazon.ConfigService.Model.RecordingGroup
RoleARN        : arn:aws:iam::123456789012:role/aws-service-role/
config.amazonaws.com/AWSServiceRoleForConfig
```

- Einzelheiten zur API finden Sie unter DescribeConfigurationRecorders AWS Tools for PowerShellCmdlet-Referenz.

`Get-CFGConfigurationRecorderStatus`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGConfigurationRecorderStatus`

Tools für PowerShell

Beispiel 1: Dieses Beispiel gibt den Status der Konfigurationsrekorder zurück.

```
Get-CFGConfigurationRecorderStatus
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeConfigurationRecorderStatus AWS Tools for PowerShellCmdlet-Referenz.

`Get-CFGConformancePack`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGConformancePack`

Tools für PowerShell

Beispiel 1: In diesem Beispiel sind alle Conformance Packs aufgeführt.

```
Get-CFGConformancePack
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeConformancePacks AWS Tools for PowerShellCmdlet-Referenz.

`Get-CFGDeliveryChannel`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGDeliveryChannel`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der Lieferkanal für die Region abgerufen und Details angezeigt.

```
Get-CFGDeliveryChannel -Region eu-west-1 | Select-Object Name, S3BucketName,
 S3KeyPrefix,
 @{N="DeliveryFrequency";E={$_.ConfigSnapshotDeliveryProperties.DeliveryFrequency}}
```

Ausgabe:

```
Name    S3BucketName               S3KeyPrefix DeliveryFrequency
----    ------------               ----------- -----------------
default config-bucket-NA my          TwentyFour_Hours
```

- Einzelheiten zur API finden Sie unter DescribeDeliveryChannels AWS Tools for PowerShellCmdlet-Referenz.

`Get-CFGResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGResourceTag`

Tools für PowerShell

Beispiel 1: Dieses Beispiel listet die zugehörigen Tags für die angegebene Ressource auf

```
Get-CFGResourceTag -ResourceArn $rules[0].ConfigRuleArn
```

Ausgabe:

```
Key     Value
---     -----
Version 1.3
```

- Einzelheiten zur API finden Sie unter ListTagsForResource AWS Tools for PowerShellCmdlet-Referenz.

`Remove-CFGConformancePack`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-CFGConformancePack`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Konformitätspaket zusammen mit allen Regeln, Behebungsmaßnahmen und Evaluierungsergebnissen für das Paket entfernt.

```
Remove-CFGConformancePack -ConformancePackName dono
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DeleteConformancePack AWS Tools for PowerShellCmdlet-Referenz.

`Write-CFGConformancePack`

Das folgende Codebeispiel zeigt die Verwendung. `Write-CFGConformancePack`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Conformance Pack erstellt, das die Vorlage aus der angegebenen Yaml-Datei abruft.

```
Write-CFGConformancePack -ConformancePackName dono -DeliveryS3Bucket kt-ps-examples
 -TemplateBody (Get-Content C:\windows\temp\template.yaml -Raw)
```

- Einzelheiten zur API finden Sie unter PutConformancePackCmdlet-Referenz.AWS Tools for PowerShell

`Write-CFGDeliveryChannel`

Das folgende Codebeispiel zeigt die Verwendung. `Write-CFGDeliveryChannel`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die DeliveryFrequency-Eigenschaft eines vorhandenen Lieferkanals geändert.

```
Write-CFGDeliveryChannel -ConfigSnapshotDeliveryProperties_DeliveryFrequency
 TwentyFour_Hours -DeliveryChannelName default -DeliveryChannel_S3BucketName config-
bucket-NA -DeliveryChannel_S3KeyPrefix my
```

- Einzelheiten zur API finden Sie unter PutDeliveryChannel AWS Tools for PowerShellCmdlet-Referenz.

Beispiele für Device Farm mit Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS Tools for PowerShell with Device Farm Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`New-DFUpload`

Das folgende Codebeispiel zeigt, wie Sie es verwenden`New-DFUpload`.

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein AWS Device Farm Farm-Upload für eine Android-App erstellt. Sie können den Projekt-ARN aus der Ausgabe von New-DFProject oder Get-DF abrufen. ProjectList Verwenden Sie die signierte URL in der New-DFUpload-Ausgabe, um eine Datei auf Device Farm hochzuladen.

```
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
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz. CreateUploadAWS Tools for PowerShell

AWS Directory Service Beispiele für die Verwendung von Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS Tools for PowerShell with Aktionen ausführen und allgemeine Szenarien implementieren AWS Directory Service.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`Add-DSIpRoute`

Das folgende Codebeispiel zeigt, wie Sie es verwendenAdd-DSIpRoute.

Tools für PowerShell

Beispiel 1: Dieser Befehl entfernt das Resource Tag, das der angegebenen Directory-ID zugewiesen ist

```
Add-DSIpRoute -DirectoryId d-123456ijkl -IpRoute @{CidrIp ="203.0.113.5/32"} -
UpdateSecurityGroupForDirectoryController $true
```

- Einzelheiten zur API finden Sie unter AddIpRoutes AWS Tools for PowerShellCmdlet-Referenz.

`Add-DSResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. Add-DSResourceTag

Tools für PowerShell

Beispiel 1: Dieser Befehl fügt der angegebenen Directory-ID das Resource-Tag hinzu

```
Add-DSResourceTag -ResourceId d-123456ijkl -Tag @{Key="myTag"; Value="mytgValue"}
```

- Einzelheiten zur API finden Sie unter AddTagsToResource AWS Tools for PowerShellCmdlet-Referenz.

`Approve-DSTrust`

Das folgende Codebeispiel zeigt die Verwendung. Approve-DSTrust

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der AWS Directory Service VerifyTrust API-Vorgang für die angegebene Trustid aufgerufen.

```
Approve-DSTrust -TrustId t-9067157123
```

- Einzelheiten zur API finden Sie unter VerifyTrust AWS Tools for PowerShellCmdlet-Referenz.

`Confirm-DSSharedDirectory`

Das folgende Codebeispiel zeigt die Verwendung. `Confirm-DSSharedDirectory`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine vom Verzeichnisbesitzer gesendete Anfrage zur gemeinsamen Nutzung von Verzeichnissen akzeptiert AWS-Konto.

```
Confirm-DSSharedDirectory -SharedDirectoryId d-9067012345
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter AcceptSharedDirectory AWS Tools for PowerShellCmdlet-Referenz.

`Connect-DSDirectory`

Das folgende Codebeispiel zeigt die Verwendung. `Connect-DSDirectory`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein AD Connector erstellt, um eine Verbindung zu einem lokalen Verzeichnis herzustellen.

```
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
```

- Einzelheiten zur API finden Sie unter ConnectDirectory AWS Tools for PowerShellCmdlet-Referenz.

`Deny-DSSharedDirectory`

Das folgende Codebeispiel zeigt die Verwendung. `Deny-DSSharedDirectory`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Anfrage zur gemeinsamen Nutzung von Verzeichnissen abgelehnt, die vom Konto des Verzeichnisbesitzers gesendet wurde.

```
Deny-DSSharedDirectory -SharedDirectoryId d-9067012345
```

Ausgabe:

```
d-9067012345
```

- Einzelheiten zur API finden Sie unter RejectSharedDirectory AWS Tools for PowerShellCmdlet-Referenz.

`Disable-DSDirectoryShare`

Das folgende Codebeispiel zeigt die Verwendung. `Disable-DSDirectoryShare`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die gemeinsame Nutzung des Verzeichnisses zwischen dem Verzeichnisbesitzer und dem Benutzerkonto beendet.

```
Disable-DSDirectoryShare -DirectoryId d-123456ijkl -UnshareTarget_Id 123456784321 -
UnshareTarget_Type ACCOUNT
```

Ausgabe:

```
d-9067012345
```

- Einzelheiten zur API finden Sie unter UnshareDirectory AWS Tools for PowerShellCmdlet-Referenz.

`Disable-DSLDAPS`

Das folgende Codebeispiel zeigt die Verwendung. `Disable-DSLDAPS`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden sichere LDAP-Aufrufe für das angegebene Verzeichnis deaktiviert.

```
Disable-DSLDAPS -DirectoryId d-123456ijkl -Type Client
```

- Einzelheiten zur API finden Sie unter DisableLDAPS in der Cmdlet-Referenz.AWS Tools for PowerShell

`Disable-DSRadius`

Das folgende Codebeispiel zeigt die Verwendung. `Disable-DSRadius`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der RADIUS-Server deaktiviert, der für einen AD Connector oder ein Microsoft AD-Verzeichnis konfiguriert ist.

```
Disable-DSRadius -DirectoryId d-123456ijkl
```

- Einzelheiten zur API finden Sie unter DisableRadius AWS Tools for PowerShellCmdlet-Referenz.

`Disable-DSSso`

Das folgende Codebeispiel zeigt die Verwendung. `Disable-DSSso`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird Single Sign-On für ein Verzeichnis deaktiviert.

```
Disable-DSSso -DirectoryId d-123456ijkl
```

- Einzelheiten zur API finden Sie unter DisableSso AWS Tools for PowerShellCmdlet-Referenz.

`Enable-DSDirectoryShare`

Das folgende Codebeispiel zeigt die Verwendung. `Enable-DSDirectoryShare`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein bestimmtes Verzeichnis in Ihrem AWS Konto mithilfe der Handshake-Methode für ein anderes AWS Konto freigegeben.

```
Enable-DSDirectoryShare -DirectoryId d-123456ijkl -ShareTarget_Id 123456784321 -
ShareMethod HANDSHAKE -ShareTarget_Type ACCOUNT
```

Ausgabe:

```
d-9067012345
```

- Einzelheiten zur API finden Sie unter ShareDirectory AWS Tools for PowerShellCmdlet-Referenz.

`Enable-DSLDAPS`

Das folgende Codebeispiel zeigt die Verwendung. `Enable-DSLDAPS`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der Switch für das spezifische Verzeichnis aktiviert, sodass immer sichere LDAP-Aufrufe verwendet werden.

```
Enable-DSLDAPS -DirectoryId d-123456ijkl -Type Client
```

- Einzelheiten zur API finden Sie unter EnableLDAPS in der Cmdlet-Referenz.AWS Tools for PowerShell

`Enable-DSRadius`

Das folgende Codebeispiel zeigt die Verwendung. `Enable-DSRadius`

Tools für PowerShell

Beispiel 1: Dieses Beispiel ermöglicht die Multi-Faktor-Authentifizierung (MFA) mit der bereitgestellten RADIUS-Serverkonfiguration für einen AD Connector oder ein Microsoft AD-Verzeichnis.

```
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
```

- Einzelheiten zur API finden Sie unter EnableRadius AWS Tools for PowerShellCmdlet-Referenz.

`Enable-DSSso`

Das folgende Codebeispiel zeigt die Verwendung. `Enable-DSSso`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird Single Sign-On für ein Verzeichnis aktiviert.

```
Enable-DSSso -DirectoryId d-123456ijkl
```

- Einzelheiten zur API finden Sie unter EnableSso AWS Tools for PowerShellCmdlet-Referenz.

`Get-DSCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSCertificate`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Informationen über das Zertifikat angezeigt, das für eine sichere LDAP-Verbindung registriert ist.

```
Get-DSCertificate -DirectoryId d-123456ijkl -CertificateId c-906731e34f
```

Ausgabe:

```
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.
```

- Einzelheiten zur API finden Sie unter DescribeCertificate AWS Tools for PowerShellCmdlet-Referenz.

`Get-DSCertificateList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSCertificateList`

Tools für PowerShell

Beispiel 1: Dieses Beispiel listet alle Zertifikate auf, die für eine sichere LDAP-Verbindung für das angegebene Verzeichnis registriert sind.

```
Get-DSCertificateList -DirectoryId d-123456ijkl
```

Ausgabe:

```
CertificateId CommonName                ExpiryDateTime       State
------------- ----------                --------------       -----
c-906731e34f  contoso-EC2AMAZ-CTGG2NM-CA 4/15/2025 6:34:15 PM Registered
```

- Einzelheiten zur API finden Sie unter ListCertificates AWS Tools for PowerShellCmdlet-Referenz.

`Get-DSConditionalForwarder`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSConditionalForwarder`

Tools für PowerShell

Beispiel 1: Dieser Befehl ruft alle konfigurierten Conditional Forwarder der angegebenen Directory-ID ab.

```
Get-DSConditionalForwarder -DirectoryId d-123456ijkl
```

Ausgabe:

```
DnsIpAddrs      RemoteDomainName ReplicationScope
----------      ---------------- ----------------
{172.31.77.239} contoso.com      Domain
```

- Einzelheiten zur API finden Sie unter DescribeConditionalForwardersCmdlet-Referenz.AWS Tools for PowerShell

`Get-DSDirectory`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSDirectory`

Tools für PowerShell

Beispiel 1: Dieser Befehl ruft Informationen über die Verzeichnisse ab, die zu diesem Konto gehören.

```
Get-DSDirectory | Select-Object DirectoryId, Name, DnsIpAddrs, Type
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeDirectories AWS Tools for PowerShellCmdlet-Referenz.

`Get-DSDirectoryLimit`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSDirectoryLimit`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Informationen zum Verzeichnislimit für die Region us-east-1 angezeigt.

```
Get-DSDirectoryLimit -Region us-east-1
```

Ausgabe:

```
CloudOnlyDirectoriesCurrentCount : 1
CloudOnlyDirectoriesLimit        : 10
CloudOnlyDirectoriesLimitReached : False
CloudOnlyMicrosoftADCurrentCount : 1
CloudOnlyMicrosoftADLimit        : 20
CloudOnlyMicrosoftADLimitReached : False
ConnectedDirectoriesCurrentCount : 1
ConnectedDirectoriesLimit        : 10
```

- Einzelheiten zur API finden Sie unter GetDirectoryLimitsCmdlet-Referenz.AWS Tools for PowerShell

`Get-DSDomainControllerList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSDomainControllerList`

Tools für PowerShell

Beispiel 1: Dieser Befehl ruft die detaillierte Liste der Domänencontroller ab, die für die angegebene Verzeichnis-ID gestartet wurden

```
Get-DSDomainControllerList -DirectoryId d-123456ijkl
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeDomainControllers AWS Tools for PowerShellCmdlet-Referenz.

`Get-DSEventTopic`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSEventTopic`

Tools für PowerShell

Beispiel 1: Dieser Befehl zeigt Informationen zum konfigurierten SNS-Thema an, sodass Sie benachrichtigt werden können, wenn sich der Verzeichnisstatus ändert.

```
Get-DSEventTopic -DirectoryId d-123456ijkl
```

Ausgabe:

```
CreatedDateTime : 12/13/2019 11:15:32 AM
DirectoryId     : d-123456ijkl
Status          : Registered
TopicArn        : arn:aws:sns:us-east-1:123456781234:snstopicname
TopicName       : snstopicname
```

- Einzelheiten zur API finden Sie unter DescribeEventTopics AWS Tools for PowerShellCmdlet-Referenz.

`Get-DSIpRouteList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSIpRouteList`

Tools für PowerShell

Beispiel 1: Mit diesem Befehl werden die in Directory IP Routing konfigurierten öffentlichen IP-Adressblöcke abgerufen

```
Get-DSIpRouteList -DirectoryId d-123456ijkl
```

Ausgabe:

```
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 :
```

- Einzelheiten zur API finden Sie unter ListIpRoutes AWS Tools for PowerShellCmdlet-Referenz.

`Get-DSLDAPSSetting`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSLDAPSSetting`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt den Status der LDAP-Sicherheit für das angegebene Verzeichnis.

```
Get-DSLDAPSSetting -DirectoryId d-123456ijkl
```

Ausgabe:

```
LastUpdatedDateTime  LDAPSStatus LDAPSStatusReason
-------------------  ----------- -----------------
4/15/2020 6:51:03 PM Enabled     LDAPS is enabled successfully.
```

- Einzelheiten zur API finden Sie unter DescribeLDAPSSettings in der Cmdlet-Referenz.AWS Tools for PowerShell

`Get-DSLogSubscriptionList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSLogSubscriptionList`

Tools für PowerShell

Beispiel 1: Dieser Befehl ruft die Protokollabonnementinformationen der angegebenen Verzeichnis-ID ab

```
Get-DSLogSubscriptionList -DirectoryId d-123456ijkl
```

Ausgabe:

```
DirectoryId  LogGroupName
 SubscriptionCreatedDateTime
-----------  ------------
 ---------------------------
d-123456ijkl /aws/directoryservice/d-123456ijkl-lan2.example.com 12/14/2019 9:05:23
 AM
```

- Einzelheiten zur API finden Sie unter ListLogSubscriptions AWS Tools for PowerShellCmdlet-Referenz.

`Get-DSResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSResourceTag`

Tools für PowerShell

Beispiel 1: Dieser Befehl ruft alle Tags des angegebenen Verzeichnisses ab.

```
Get-DSResourceTag -ResourceId d-123456ijkl
```

Ausgabe:

```
Key   Value
---   -----
myTag myTagValue
```

- Einzelheiten zur API finden Sie unter ListTagsForResource AWS Tools for PowerShellCmdlet-Referenz.

`Get-DSSchemaExtension`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSSchemaExtension`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden alle Schemaerweiterungen aufgeführt, die auf ein Microsoft AD-Verzeichnis angewendet wurden.

```
Get-DSSchemaExtension -DirectoryId d-123456ijkl
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter ListSchemaExtensions AWS Tools for PowerShellCmdlet-Referenz.

`Get-DSSharedDirectory`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSSharedDirectory`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die geteilten Verzeichnisse Ihres AWS Kontos abgerufen

```
Get-DSSharedDirectory -OwnerDirectoryId d-123456ijkl -SharedDirectoryId d-9067012345
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeSharedDirectories AWS Tools for PowerShellCmdlet-Referenz.

`Get-DSSnapshot`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSSnapshot`

Tools für PowerShell

Beispiel 1: Mit diesem Befehl werden Informationen zu den angegebenen Verzeichnis-Snapshots abgerufen, die zu diesem Konto gehören.

```
Get-DSSnapshot -DirectoryId d-123456ijkl
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeSnapshots AWS Tools for PowerShellCmdlet-Referenz.

`Get-DSSnapshotLimit`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSSnapshotLimit`

Tools für PowerShell

Beispiel 1: Mit diesem Befehl werden die manuellen Snapshot-Grenzwerte für ein bestimmtes Verzeichnis abgerufen.

```
Get-DSSnapshotLimit -DirectoryId d-123456ijkl
```

Ausgabe:

```
ManualSnapshotsCurrentCount ManualSnapshotsLimit ManualSnapshotsLimitReached
--------------------------- -------------------- --------------------------
0                           5                    False
```

- Einzelheiten zur API finden Sie unter GetSnapshotLimits AWS Tools for PowerShellCmdlet-Referenz.

`Get-DSTrust`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSTrust`

Tools für PowerShell

Beispiel 1: Mit diesem Befehl werden Informationen zu Vertrauensbeziehungen abgerufen, die für die angegebene Verzeichnis-ID erstellt wurden.

```
Get-DSTrust -DirectoryId d-123456abcd
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeTrusts AWS Tools for PowerShellCmdlet-Referenz.

`New-DSAlias`

Das folgende Codebeispiel zeigt die Verwendung. `New-DSAlias`

Tools für PowerShell

Beispiel 1: Dieser Befehl erstellt einen Alias für ein Verzeichnis und weist den Alias der angegebenen Verzeichnis-ID zu.

```
New-DSAlias -DirectoryId d-123456ijkl -Alias MyOrgName
```

Ausgabe:

```
Alias     DirectoryId
-----     -----------
myorgname d-123456ijkl
```

- Einzelheiten zur API finden Sie unter CreateAliasCmdlet-Referenz.AWS Tools for PowerShell

`New-DSComputer`

Das folgende Codebeispiel zeigt die Verwendung. `New-DSComputer`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein neues Active Directory-Computerobjekt erstellt.

```
New-DSComputer -DirectoryId d-123456ijkl -ComputerName ADMemberServer -Password
 $Password
```

Ausgabe:

```
ComputerAttributes                ComputerId
 ComputerName
```

```
------------------                    ----------
 ------------
{WindowsSamName, DistinguishedName} S-1-5-21-1191241402-978882507-2717148213-1662
 ADMemberServer
```

- Einzelheiten zur API finden Sie unter CreateComputer AWS Tools for PowerShellCmdlet-Referenz.

`New-DSConditionalForwarder`

Das folgende Codebeispiel zeigt die Verwendung. `New-DSConditionalForwarder`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Conditional Forwarder in der angegebenen AWS Directory-ID erstellt.

```
New-DSConditionalForwarder -DirectoryId d-123456ijkl -DnsIpAddr
 172.31.36.96,172.31.10.56 -RemoteDomainName contoso.com
```

- Einzelheiten zur API finden Sie unter CreateConditionalForwarderCmdlet-Referenz.AWS Tools for PowerShell

`New-DSDirectory`

Das folgende Codebeispiel zeigt die Verwendung. `New-DSDirectory`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein neues Simple AD AD-Verzeichnis erstellt.

```
New-DSDirectory -Name corp.example.com -Password $Password -Size Small -
VpcSettings_VpcId vpc-123459d -VpcSettings_SubnetIds subnet-1234kkaa,subnet-5678ffbb
```

- Einzelheiten zur API finden Sie unter CreateDirectory AWS Tools for PowerShellCmdlet-Referenz.

`New-DSLogSubscription`

Das folgende Codebeispiel zeigt die Verwendung. `New-DSLogSubscription`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Abonnement für die Weiterleitung von Directory-Service-Domänencontroller-Sicherheitsprotokollen in Echtzeit an die angegebene CloudWatch Amazon-Protokollgruppe in Ihrem erstellt AWS-Konto.

```
New-DSLogSubscription -DirectoryId d-123456ijkl -LogGroupName /aws/directoryservice/
d-123456ijkl-lan2.example.com
```

- Einzelheiten zur API finden Sie unter CreateLogSubscription AWS Tools for PowerShellCmdlet-Referenz.

`New-DSMicrosoftAD`

Das folgende Codebeispiel zeigt die Verwendung. `New-DSMicrosoftAD`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein neues Microsoft AD-Verzeichnis in erstellt AWS Cloud.

```
New-DSMicrosoftAD -Name corp.example.com -Password $Password -edition Standard -
VpcSettings_VpcId vpc-123459d -VpcSettings_SubnetIds subnet-1234kkaa,subnet-5678ffbb
```

- API-Einzelheiten finden Sie unter CreateMicrosoftAD in der AWS Tools for PowerShell Cmdlet-Referenz.

`New-DSSnapshot`

Das folgende Codebeispiel zeigt die Verwendung. `New-DSSnapshot`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Verzeichnis-Snapshot erstellt

```
New-DSSnapshot -DirectoryId d-123456ijkl
```

- Einzelheiten zur API finden Sie unter CreateSnapshot AWS Tools for PowerShellCmdlet-Referenz.

`New-DSTrust`

Das folgende Codebeispiel zeigt die Verwendung. `New-DSTrust`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine bidirektionale Forestweite Vertrauensstellung zwischen Ihrem AWS verwalteten Microsoft AD-Verzeichnis und dem vorhandenen lokalen Microsoft Active Directory hergestellt.

```
New-DSTrust -DirectoryId d-123456ijkl -RemoteDomainName contoso.com -TrustDirection
 Two-Way -TrustType Forest -TrustPassword $Password -ConditionalForwarderIpAddr
 172.31.36.96
```

Ausgabe:

```
t-9067157123
```

- Einzelheiten zur API finden Sie unter CreateTrustCmdlet-Referenz.AWS Tools for PowerShell

`Register-DSCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `Register-DSCertificate`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Zertifikat für eine sichere LDAP-Verbindung registriert.

```
$Certificate = Get-Content contoso.cer -Raw
Register-DSCertificate -DirectoryId d-123456ijkl -CertificateData $Certificate
```

Ausgabe:

```
c-906731e350
```

- Einzelheiten zur API finden Sie unter RegisterCertificate AWS Tools for PowerShellCmdlet-Referenz.

`Register-DSEventTopic`

Das folgende Codebeispiel zeigt die Verwendung. `Register-DSEventTopic`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Verzeichnis als Herausgeber einem SNS-Thema zugeordnet.

```
Register-DSEventTopic -DirectoryId d-123456ijkl -TopicName snstopicname
```

- Einzelheiten zur API finden Sie unter RegisterEventTopic AWS Tools for PowerShellCmdlet-Referenz.

`Remove-DSConditionalForwarder`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-DSConditionalForwarder`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die bedingte Weiterleitung entfernt, die für Ihr AWS Verzeichnis eingerichtet wurde.

```
Remove-DSConditionalForwarder -DirectoryId d-123456ijkl -RemoteDomainName
 contoso.com
```

- Einzelheiten zur API finden Sie unter DeleteConditionalForwarderCmdlet-Referenz.AWS Tools for PowerShell

`Remove-DSDirectory`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-DSDirectory`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein AWS Verzeichnisdienstverzeichnis gelöscht (Simple AD/ Microsoft AD/AD Connector)

```
Remove-DSDirectory -DirectoryId d-123456ijkl
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz. DeleteDirectoryAWS Tools for PowerShell

`Remove-DSIpRoute`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-DSIpRoute`

Tools für PowerShell

Beispiel 1: Dieser Befehl entfernt die angegebene IP aus den konfigurierten IP-Routen von Directory-ID.

```
Remove-DSIpRoute -DirectoryId d-123456ijkl -CidrIp 203.0.113.5/32
```

- Einzelheiten zur API finden Sie unter RemoveIpRoutes AWS Tools for PowerShellCmdlet-Referenz.

`Remove-DSLogSubscription`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-DSLogSubscription`

Tools für PowerShell

Beispiel 1: Dieser Befehl entfernt das Protokollabonnement der angegebenen Verzeichnis-ID

```
Remove-DSLogSubscription -DirectoryId d-123456ijkl
```

- Einzelheiten zur API finden Sie unter DeleteLogSubscription AWS Tools for PowerShellCmdlet-Referenz.

`Remove-DSResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-DSResourceTag`

Tools für PowerShell

Beispiel 1: Dieser Befehl entfernt das Resource Tag, das der angegebenen Directory-ID zugewiesen ist

```
Remove-DSResourceTag -ResourceId d-123456ijkl -TagKey myTag
```

- Einzelheiten zur API finden Sie unter RemoveTagsFromResource AWS Tools for PowerShellCmdlet-Referenz.

`Remove-DSSnapshot`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-DSSnapshot`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der manuell erstellte Snapshot entfernt.

```
Remove-DSSnapshot -SnapshotId s-9068b488kc
```

- Einzelheiten zur API finden Sie unter DeleteSnapshot AWS Tools for PowerShellCmdlet-Referenz.

`Remove-DSTrust`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-DSTrust`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die bestehende Vertrauensstellung zwischen Ihrem AWS Managed AD-Verzeichnis und einer externen Domain entfernt.

```
Get-DSTrust -DirectoryId d-123456ijkl -Select Trusts.TrustId | Remove-DSTrust
```

Ausgabe:

```
t-9067157123
```

- Einzelheiten zur API finden Sie unter DeleteTrustCmdlet-Referenz.AWS Tools for PowerShell

`Reset-DSUserPassword`

Das folgende Codebeispiel zeigt die Verwendung. `Reset-DSUserPassword`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Passwort eines Active Directory-Benutzers namens ADUser in AWS Managed Microsoft AD oder Simple AD Directory zurückgesetzt

```
Reset-DSUserPassword -UserName ADuser -DirectoryId d-123456ijkl -NewPassword
 $Password
```

- Einzelheiten zur API finden Sie unter ResetUserPasswordCmdlet-Referenz.AWS Tools for PowerShell

`Restore-DSFromSnapshot`

Das folgende Codebeispiel zeigt die Verwendung. `Restore-DSFromSnapshot`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Verzeichnis mithilfe eines vorhandenen Verzeichnis-Snapshots wiederhergestellt.

```
Restore-DSFromSnapshot -SnapshotId s-9068b488kc
```

- Einzelheiten zur API finden Sie unter RestoreFromSnapshot AWS Tools for PowerShellCmdlet-Referenz.

`Set-DSDomainControllerCount`

Das folgende Codebeispiel zeigt die Verwendung. `Set-DSDomainControllerCount`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Anzahl der Domänencontroller für die angegebene Verzeichnis-ID auf 3 gesetzt.

```
Set-DSDomainControllerCount -DirectoryId d-123456ijkl -DesiredNumber 3
```

- Einzelheiten zur API finden Sie unter UpdateNumberOfDomainControllers AWS Tools for PowerShellCmdlet-Referenz.

`Start-DSSchemaExtension`

Das folgende Codebeispiel zeigt die Verwendung. `Start-DSSchemaExtension`

Tools für PowerShell

Beispiel 1: Dieses Beispiel wendet eine Schemaerweiterung auf ein Microsoft AD-Verzeichnis an.

```
$ldif = Get-Content D:\Users\Username\Downloads\ExtendedSchema.ldf -Raw
```

```
Start-DSSchemaExtension -DirectoryId d-123456ijkl -
CreateSnapshotBeforeSchemaExtension $true -Description ManagedADSchemaExtension -
LdifContent $ldif
```

Ausgabe:

```
e-9067306643
```

- Einzelheiten zur API finden Sie unter StartSchemaExtension AWS Tools for PowerShellCmdlet-Referenz.

`Stop-DSSchemaExtension`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-DSSchemaExtension`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine in Bearbeitung befindliche Schemaerweiterung für ein Microsoft AD-Verzeichnis storniert.

```
Stop-DSSchemaExtension -DirectoryId d-123456ijkl -SchemaExtensionId e-9067306643
```

- Einzelheiten zur API finden Sie unter CancelSchemaExtension AWS Tools for PowerShellCmdlet-Referenz.

`Unregister-DSCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-DSCertificate`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Zertifikat, das für eine sichere LDAP-Verbindung registriert wurde, aus dem System gelöscht.

```
Unregister-DSCertificate -DirectoryId d-123456ijkl -CertificateId c-906731e34f
```

- Einzelheiten zur API finden Sie unter DeregisterCertificate AWS Tools for PowerShellCmdlet-Referenz.

`Unregister-DSEventTopic`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-DSEventTopic`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Verzeichnis als Herausgeber für das angegebene SNS-Thema entfernt.

```
Unregister-DSEventTopic -DirectoryId d-123456ijkl -TopicName snstopicname
```

- Einzelheiten zur API finden Sie unter DeregisterEventTopicCmdlet-Referenz.AWS Tools for PowerShell

`Update-DSConditionalForwarder`

Das folgende Codebeispiel zeigt die Verwendung. `Update-DSConditionalForwarder`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine bedingte Weiterleitung aktualisiert, die für Ihr AWS Verzeichnis eingerichtet wurde.

```
Update-DSConditionalForwarder -DirectoryId d-123456ijkl -DnsIpAddr
 172.31.36.96,172.31.16.108 -RemoteDomainName contoso.com
```

- Einzelheiten zur API finden Sie unter UpdateConditionalForwarder AWS Tools for PowerShellCmdlet-Referenz.

`Update-DSRadius`

Das folgende Codebeispiel zeigt die Verwendung. `Update-DSRadius`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden RADIUS-Serverinformationen für einen AD Connector oder ein Microsoft AD-Verzeichnis aktualisiert.

```
Update-DSRadius -DirectoryId d-123456ijkl -RadiusSettings_RadiusRetry 3
```

- Einzelheiten zur API finden Sie unter UpdateRadius AWS Tools for PowerShellCmdlet-Referenz.

`Update-DSTrust`

Das folgende Codebeispiel zeigt die Verwendung. `Update-DSTrust`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der SelectiveAuth Parameter der angegebenen Trust-ID von Disabled auf Enabled aktualisiert.

```
Update-DSTrust -TrustId t-9067157123 -SelectiveAuth Enabled
```

Ausgabe:

```
RequestId                            TrustId
---------                            -------
138864a7-c9a8-4ad1-a828-eae479e85b45 t-9067157123
```

- Einzelheiten zur API finden Sie unter UpdateTrust AWS Tools for PowerShellCmdlet-Referenz.

AWS DMS Beispiele für die Verwendung von Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS Tools for PowerShell with Aktionen ausführen und allgemeine Szenarien implementieren AWS DMS.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`New-DMSReplicationTask`

Das folgende Codebeispiel zeigt, wie Sie es verwendenNew-DMSReplicationTask.

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine neue Replikationsaufgabe des AWS Datenbankmigrationsdienstes erstellt, die CdcStartTime anstelle von verwendet CdcStartPosition. Der MigrationType ist auf "full-load-and-cdc" gesetzt, was bedeutet, dass die Zieltabelle leer sein muss. Die neue Aufgabe ist mit einem Tag gekennzeichnet, das den Schlüssel Stage und den Schlüsselwert Test hat. Weitere Informationen zu den von diesem Cmdlet verwendeten Werten finden Sie unter Creating a Task (https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Tasks.Creating.html) im AWS Database Migration Service Service-Benutzerhandbuch.

```
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"
```

- Einzelheiten zur API finden Sie unter CreateReplicationTask AWS Tools for PowerShellCmdlet-Referenz.

DynamoDB-Beispiele mit Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS Tools for PowerShell mit DynamoDB Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

Aktionen

`Add-DDBIndexSchema`

Das folgende Codebeispiel zeigt die Verwendung`Add-DDBIndexSchema`.

Tools für PowerShell

Beispiel 1: Erstellt ein leeres TableSchema Objekt und fügt ihm eine neue Definition für den lokalen sekundären Index hinzu, bevor das TableSchema Objekt in die Pipeline geschrieben wird.

```
$schema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
 "LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
$schema = New-DDBTableSchema
```

Ausgabe:

```
AttributeSchema                             KeySchema
    LocalSecondaryIndexSchema
---------------                             ---------
    -------------------------
{LastPostDateTime}                          {}
    {LastPostIndex}
```

Beispiel 2: Fügt dem bereitgestellten TableSchema Objekt eine neue lokale sekundäre Indexdefinition hinzu, bevor das TableSchema Objekt wieder in die Pipeline geschrieben wird. Das TableSchema Objekt kann auch mit dem Parameter -Schema bereitgestellt werden.

```
New-DDBTableSchema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
 "LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
```

Ausgabe:

```
AttributeSchema                             KeySchema
    LocalSecondaryIndexSchema
---------------                             ---------
    -------------------------
{LastPostDateTime}                          {}
    {LastPostIndex}
```

- Einzelheiten zur API finden Sie unter Add-DDB IndexSchema in AWS Tools for PowerShell der Cmdlet-Referenz.

`Add-DDBKeySchema`

Das folgende Codebeispiel zeigt die Verwendung. `Add-DDBKeySchema`

Tools für PowerShell

Beispiel 1: Erstellt ein leeres TableSchema Objekt und fügt ihm anhand der angegebenen Schlüsseldaten Einträge für Schlüssel- und Attributdefinitionen hinzu, bevor das TableSchema Objekt in die Pipeline geschrieben wird. Der Schlüsseltyp ist standardmäßig als 'HASH' deklariert. Verwenden Sie den KeyType Parameter - mit dem Wert 'RANGE', um einen Bereichsschlüssel zu deklarieren.

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
```

Ausgabe:

```
AttributeSchema                             KeySchema
    LocalSecondaryIndexSchema
---------------                             ---------
    -------------------------
{ForumName}                                 {ForumName}
    {}
```

Beispiel 2: Fügt dem bereitgestellten TableSchema Objekt neue Schlüssel- und Attributdefinitionseinträge hinzu, bevor das Objekt in die TableSchema Pipeline geschrieben wird. Der Schlüsseltyp ist standardmäßig als 'HASH' deklariert. Verwenden Sie den KeyType

Parameter - mit dem Wert 'RANGE', um einen Bereichsschlüssel zu deklarieren. Das TableSchema Objekt kann auch mit dem Parameter -Schema angegeben werden.

```
New-DDBTableSchema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
```

Ausgabe:

```
AttributeSchema                             KeySchema
    LocalSecondaryIndexSchema
---------------                             ---------
    -------------------------
{ForumName}                                 {ForumName}
    {}
```

- Einzelheiten zur API finden Sie unter Add-DDB KeySchema in AWS Tools for PowerShell der Cmdlet-Referenz.

`ConvertFrom-DDBItem`

Das folgende Codebeispiel zeigt die Verwendung. `ConvertFrom-DDBItem`

Tools für PowerShell

Beispiel 1: ConvertFrom -DDBItem wird verwendet, um das Ergebnis von Get-DDBItem aus einer Hashtabelle von DynamoDB in eine Hashtabelle mit gängigen Typen wie string und double AttributeValues zu konvertieren.

```
@{
    SongTitle = 'Somewhere Down The Road'
    Artist    = 'No One You Know'
} | ConvertTo-DDBItem

Get-DDBItem -TableName 'Music' -Key $key | ConvertFrom-DDBItem
```

Ausgabe:

```
Name                           Value
----                           -----
Genre                          Country
Artist                         No One You Know
Price                          1.94
```

```
CriticRating                    9
SongTitle                       Somewhere Down The Road
AlbumTitle                      Somewhat Famous
```

- Einzelheiten zur API finden Sie unter -DDBItem in der Cmdlet-Referenz. ConvertFrom AWS Tools for PowerShell

`ConvertTo-DDBItem`

Das folgende Codebeispiel zeigt die Verwendung. `ConvertTo-DDBItem`

Tools für PowerShell

Beispiel 1: Ein Beispiel für die Konvertierung einer Hashtabelle in ein Wörterbuch mit DynamoDB-Attributwerten.

```
@{
    SongTitle = 'Somewhere Down The Road'
    Artist    = 'No One You Know'
} | ConvertTo-DDBItem

Key       Value
---       -----
SongTitle Amazon.DynamoDBv2.Model.AttributeValue
Artist    Amazon.DynamoDBv2.Model.AttributeValue
```

Beispiel 2: Ein Beispiel für die Konvertierung einer Hashtabelle in ein Wörterbuch mit DynamoDB-Attributwerten.

```
@{
    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
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter ConvertTo-DDBItem in der Cmdlet-Referenz.AWS Tools for PowerShell

`Get-DDBBatchItem`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DDBBatchItem`

Tools für PowerShell

Beispiel 1: Ruft das Element mit dem Namen SongTitle „Somewhere Down The Road“ aus den DynamoDB-Tabellen „Music“ und „Songs“ ab.

```
$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
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz. BatchGetItemAWS Tools for PowerShell

`Get-DDBItem`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DDBItem`

Tools für PowerShell

Beispiel 1: Gibt das DynamoDB-Element mit dem Partitionsschlüssel SongTitle und dem Sortierschlüssel Artist zurück.

```
$key = @{
  SongTitle = 'Somewhere Down The Road'
  Artist = 'No One You Know'
} | ConvertTo-DDBItem

Get-DDBItem -TableName 'Music' -Key $key | ConvertFrom-DDBItem
```

Ausgabe:

```
Name                           Value
----                           -----
Genre                          Country
SongTitle                      Somewhere Down The Road
Price                          1.94
Artist                         No One You Know
CriticRating                   9
AlbumTitle                     Somewhat Famous
```

- Einzelheiten zur API finden Sie unter GetItem AWS Tools for PowerShellCmdlet-Referenz.

`Get-DDBTable`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DDBTable`

Tools für PowerShell

Beispiel 1: Gibt Details der angegebenen Tabelle zurück.

```
Get-DDBTable -TableName "myTable"
```

- Einzelheiten zur API finden Sie unter DescribeTable AWS Tools for PowerShellCmdlet-Referenz.

`Get-DDBTableList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DDBTableList`

Tools für PowerShell

Beispiel 1: Gibt Details aller Tabellen zurück und iteriert automatisch, bis der Service anzeigt, dass keine weiteren Tabellen existieren.

```
Get-DDBTableList
```

Beispiel 2: Iteriert manuell nach Details aller Tabellen und gibt bis zu 10 Tabellen pro Aufruf zurück, bis der Service feststellt, dass keine weiteren Tabellen existieren.

```
$nextToken = $null
do {
```

```
  Get-DDBTableList -ExclusiveStartTableName $nextToken -Limit 10
  $nextToken = $AWSHistory.LastServiceResponse.LastEvaluatedTableName
} while ($nextToken -ne $null)
```

- Einzelheiten zur API finden Sie unter ListTables AWS Tools for PowerShellCmdlet-Referenz.

`Invoke-DDBQuery`

Das folgende Codebeispiel zeigt die Verwendung. `Invoke-DDBQuery`

Tools für PowerShell

Beispiel 1: Ruft eine Abfrage auf, die DynamoDB-Elemente mit dem angegebenen SongTitle Wert und Artist zurückgibt.

```
$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
```

Ausgabe:

```
Name                           Value
----                           -----
Genre                          Country
Artist                         No One You Know
Price                          1.94
CriticRating                   9
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
```

- Einzelheiten zur API finden Sie unter Query in AWS Tools for PowerShell Cmdlet Reference.

`Invoke-DDBScan`

Das folgende Codebeispiel zeigt die Verwendung. `Invoke-DDBScan`

Beispiel 1: Gibt alle Elemente in der Tabelle Musik zurück.

```
Invoke-DDBScan -TableName 'Music' | ConvertFrom-DDBItem
```

Ausgabe:

```
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
```

Beispiel 2: Gibt Elemente in der Tabelle Musik zurück, deren Wert CriticRating größer oder gleich neun ist.

```
$scanFilter = @{
        CriticRating = [Amazon.DynamoDBv2.Model.Condition]@{
            AttributeValueList = @(@{N = '9'})
            ComparisonOperator = 'GE'
        }
    }
    Invoke-DDBScan -TableName 'Music' -ScanFilter $scanFilter | ConvertFrom-DDBItem
```

Ausgabe:

```
Name                           Value
----                           -----
Genre                          Country
Artist                         No One You Know
Price                          1.94
```

```
CriticRating                 9
SongTitle                    Somewhere Down The Road
AlbumTitle                   Somewhat Famous
```

- Einzelheiten zur API finden Sie unter Referenz zum Scannen von AWS Tools for PowerShell Cmdlets.

`New-DDBTable`

Das folgende Codebeispiel zeigt die Verwendung. `New-DDBTable`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Tabelle mit dem Namen Thread erstellt, deren Primärschlüssel aus 'ForumName' (Schlüsseltyp-Hash) und 'Subject' (Schlüsseltypbereich) besteht. Das zur Erstellung der Tabelle verwendete Schema kann wie gezeigt oder mit dem Parameter -Schema angegeben an jedes Cmdlet übergeben werden.

```
$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
```

Ausgabe:

```
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  : {}
```

Beispiel 2: In diesem Beispiel wird eine Tabelle mit dem Namen Thread erstellt, deren Primärschlüssel aus 'ForumName' (Schlüsseltyp-Hash) und 'Subject' (Schlüsseltypbereich) besteht. Ein lokaler sekundärer Index ist ebenfalls definiert. Der Schlüssel des lokalen sekundären Indexes wird automatisch anhand des primären Hashschlüssels in der Tabelle festgelegt (ForumName). Das zur Erstellung der Tabelle verwendete Schema kann über die Pipeline an

jedes Cmdlet übergeben werden, wie in der Abbildung gezeigt oder mit dem Parameter -Schema angegeben.

```
$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
```

Ausgabe:

```
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}
```

Beispiel 3: Dieses Beispiel zeigt, wie eine einzelne Pipeline verwendet wird, um eine Tabelle mit dem Namen Thread zu erstellen, deren Primärschlüssel aus 'ForumName' (Schlüsseltyp-Hash) und 'Subject' (Schlüsseltypbereich) und einem lokalen Sekundärindex besteht. Add-DDB KeySchema und Add-DDB IndexSchema erstellen ein neues TableSchema Objekt für Sie, falls keines über die Pipeline oder den Parameter -Schema bereitgestellt wird.

```
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
```

Ausgabe:

```
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}
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz. CreateTableAWS Tools for PowerShell

`New-DDBTableSchema`

Das folgende Codebeispiel zeigt die Verwendung. `New-DDBTableSchema`

Tools für PowerShell

Beispiel 1: Erstellt ein leeres TableSchema Objekt, das bereit ist, Schlüssel- und Indexdefinitionen für die Erstellung einer neuen Amazon DynamoDB-Tabelle zu akzeptieren. Das zurückgegebene Objekt kann über die Pipeline an die Cmdlets Add-DDBKeySchema, Add-DDB IndexSchema und New-DDBTable übergeben werden oder mithilfe des -Schema-Parameters in jedem Cmdlet an sie übergeben werden.

```
New-DDBTableSchema
```

Ausgabe:

```
AttributeSchema                             KeySchema
    LocalSecondaryIndexSchema
---------------                             ---------
    -------------------------
{}                                          {}
    {}
```

- Einzelheiten zur API finden Sie unter New-DDB TableSchema in AWS Tools for PowerShell der Cmdlet-Referenz.

`Remove-DDBItem`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-DDBItem`

Tools für PowerShell

Beispiel 1: Entfernt das DynamoDB-Element, das dem angegebenen Schlüssel entspricht.

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem
Remove-DDBItem -TableName 'Music' -Key $key -Confirm:$false
```

- Einzelheiten zur API finden Sie unter DeleteItem AWS Tools for PowerShellCmdlet-Referenz.

`Remove-DDBTable`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-DDBTable`

Tools für PowerShell

Beispiel 1: Löscht die angegebene Tabelle. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird.

```
Remove-DDBTable -TableName "myTable"
```

Beispiel 2: Löscht die angegebene Tabelle. Sie werden nicht zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird.

```
Remove-DDBTable -TableName "myTable" -Force
```

- Einzelheiten zur API finden Sie unter DeleteTable AWS Tools for PowerShellCmdlet-Referenz.

`Set-DDBBatchItem`

Das folgende Codebeispiel zeigt die Verwendung. `Set-DDBBatchItem`

Tools für PowerShell

Beispiel 1: Erstellt ein neues Element oder ersetzt ein vorhandenes Element durch ein neues Element in den DynamoDB-Tabellen Music und 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
```

Ausgabe:

```
$requestItem = @{
    'Music' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
    'Songs' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
}

Set-DDBBatchItem -RequestItem $requestItem
```

- Einzelheiten zur API finden Sie unter BatchWriteItem AWS Tools for PowerShellCmdlet-Referenz.

`Set-DDBItem`

Das folgende Codebeispiel zeigt die Verwendung. `Set-DDBItem`

Tools für PowerShell

Beispiel 1: Erstellt ein neues Element oder ersetzt ein vorhandenes Element durch ein neues Element.

```
$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
```

- Einzelheiten zur API finden Sie unter PutItem AWS Tools for PowerShellCmdlet-Referenz.

`Update-DDBItem`

Das folgende Codebeispiel zeigt die Verwendung. `Update-DDBItem`

Tools für PowerShell

Beispiel 1: Setzt das Genre-Attribut auf 'Rap' für das DynamoDB-Element mit dem Partitionsschlüssel SongTitle und dem Sortierschlüssel 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
```

Ausgabe:

```
Name                           Value
----                           -----
Genre                          Rap
```

- Einzelheiten zur API finden Sie unter UpdateItem AWS Tools for PowerShellCmdlet-Referenz.

`Update-DDBTable`

Das folgende Codebeispiel zeigt die Verwendung. `Update-DDBTable`

Tools für PowerShell

Beispiel 1: Aktualisiert den bereitgestellten Durchsatz für die angegebene Tabelle.

```
Update-DDBTable -TableName "myTable" -ReadCapacity 10 -WriteCapacity 5
```

- Einzelheiten zur API finden Sie unter UpdateTable AWS Tools for PowerShellCmdlet-Referenz.

Amazon EC2 EC2-Beispiele mit Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon EC2 Aktionen ausführen und allgemeine Szenarien implementieren. AWS Tools for PowerShell

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`Add-EC2CapacityReservation`

Das folgende Codebeispiel zeigt die Verwendung`Add-EC2CapacityReservation`.

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine neue Kapazitätsreservierung mit den angegebenen Attributen erstellt

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

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter CreateCapacityReservation AWS Tools for PowerShellCmdlet-Referenz.

`Add-EC2InternetGateway`

Das folgende Codebeispiel zeigt die Verwendung. `Add-EC2InternetGateway`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Internet-Gateway an die angegebene VPC angehängt.

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

Beispiel 2: In diesem Beispiel werden eine VPC und ein Internet-Gateway erstellt und anschließend das Internet-Gateway mit der VPC verbunden.

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

- Einzelheiten zur API finden Sie unter AttachInternetGatewayCmdlet-Referenz.AWS Tools for PowerShell

`Add-EC2NetworkInterface`

Das folgende Codebeispiel zeigt die Verwendung. `Add-EC2NetworkInterface`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerkschnittstelle an die angegebene Instanz angehängt.

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

Ausgabe:

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

- Einzelheiten zur API finden Sie unter AttachNetworkInterface AWS Tools for PowerShellCmdlet-Referenz.

`Add-EC2Volume`

Das folgende Codebeispiel zeigt die Verwendung. `Add-EC2Volume`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Volume an die angegebene Instanz angehängt und mit dem angegebenen Gerätenamen verfügbar gemacht.

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

Ausgabe:

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

- Einzelheiten zur API finden Sie unter AttachVolume AWS Tools for PowerShellCmdlet-Referenz.

`Add-EC2VpnGateway`

Das folgende Codebeispiel zeigt die Verwendung. `Add-EC2VpnGateway`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Virtual Private Gateway an die angegebene VPC angehängt.

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

Ausgabe:

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

- Einzelheiten zur API finden Sie unter AttachVpnGateway AWS Tools for PowerShellCmdlet-Referenz.

`Approve-EC2VpcPeeringConnection`

Das folgende Codebeispiel zeigt die Verwendung. `Approve-EC2VpcPeeringConnection`

Tools für PowerShell

Beispiel 1: Dieses Beispiel genehmigt die angeforderte VpcPeeringConnectionId Datei pcx-1dfad234b56ff78be

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

Ausgabe:

```
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
```

- Einzelheiten AcceptVpcPeeringConnectionzur API finden AWS Tools for PowerShell Sie unter Cmdlet-Referenz.

`Confirm-EC2ProductInstance`

Das folgende Codebeispiel zeigt die Verwendung. `Confirm-EC2ProductInstance`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ermittelt, ob der angegebene Produktcode der angegebenen Instanz zugeordnet ist.

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

- Einzelheiten zur API finden Sie unter ConfirmProductInstance AWS Tools for PowerShellCmdlet-Referenz.

`Copy-EC2Image`

Das folgende Codebeispiel zeigt die Verwendung. `Copy-EC2Image`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene AMI in der Region „EU (Irland)“ in die Region „USA West (Oregon)“ kopiert. Wenn -Region nicht angegeben ist, wird die aktuelle Standardregion als Zielregion verwendet.

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

Ausgabe:

```
ami-87654321
```

- Einzelheiten zur API finden Sie unter CopyImage AWS Tools for PowerShellCmdlet-Referenz.

`Copy-EC2Snapshot`

Das folgende Codebeispiel zeigt die Verwendung. `Copy-EC2Snapshot`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Snapshot aus der Region EU (Irland) in die Region USA West (Oregon) kopiert.

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

Beispiel 2: Wenn Sie eine Standardregion festlegen und den Parameter Region weglassen, ist die Standardzielregion die Standardregion.

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

- Einzelheiten zur API finden Sie unter CopySnapshot AWS Tools for PowerShellCmdlet-Referenz.

`Deny-EC2VpcPeeringConnection`

Das folgende Codebeispiel zeigt die Verwendung. `Deny-EC2VpcPeeringConnection`

Tools für PowerShell

Beispiel 1: Das obige Beispiel lehnt die Anfrage nach der Anforderungs-ID VpcPeering pcx-01a2b3ce45fe67eb8 ab

```
Deny-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-01a2b3ce45fe67eb8
```

- Einzelheiten zur RejectVpcPeeringConnectionAPI AWS Tools for PowerShell finden Sie unter Cmdlet-Referenz.

`Disable-EC2VgwRoutePropagation`

Das folgende Codebeispiel zeigt die Verwendung. `Disable-EC2VgwRoutePropagation`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird verhindert, dass das VGW Routen automatisch an die angegebene Routingtabelle weitergibt.

```
Disable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter DisableVgwRoutePropagationCmdlet-Referenz.AWS Tools for PowerShell

`Disable-EC2VpcClassicLink`

Das folgende Codebeispiel zeigt die Verwendung. `Disable-EC2VpcClassicLink`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird EC2 VpcClassicLink für den vpc-01e23c4a5d6db78e9 deaktiviert. Es gibt entweder True oder False zurück

```
Disable-EC2VpcClassicLink -VpcId vpc-01e23c4a5d6db78e9
```

- Einzelheiten zur API finden Sie unter DisableVpcClassicLink AWS Tools for PowerShellCmdlet-Referenz.

`Disable-EC2VpcClassicLinkDnsSupport`

Das folgende Codebeispiel zeigt die Verwendung. `Disable-EC2VpcClassicLinkDnsSupport`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die ClassicLink DNS-Unterstützung für die Datei vpc-0b12d3456a7e8910d deaktiviert

```
Disable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d
```

- Einzelheiten zur DisableVpcClassicLinkDnsSupportAPI finden AWS Tools for PowerShell Sie unter Cmdlet-Referenz.

`Dismount-EC2InternetGateway`

Das folgende Codebeispiel zeigt die Verwendung. `Dismount-EC2InternetGateway`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Internet-Gateway von der angegebenen VPC getrennt.

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

- Einzelheiten zur API finden Sie unter DetachInternetGateway AWS Tools for PowerShellCmdlet-Referenz.

`Dismount-EC2NetworkInterface`

Das folgende Codebeispiel zeigt die Verwendung. `Dismount-EC2NetworkInterface`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Verbindung zwischen einer Netzwerkschnittstelle und einer Instanz entfernt.

```
Dismount-EC2NetworkInterface -AttachmentId eni-attach-1a2b3c4d -Force
```

- Einzelheiten zur API finden Sie unter DetachNetworkInterface AWS Tools for PowerShellCmdlet-Referenz.

`Dismount-EC2Volume`

Das folgende Codebeispiel zeigt die Verwendung. `Dismount-EC2Volume`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Volume getrennt.

```
Dismount-EC2Volume -VolumeId vol-12345678
```

Ausgabe:

```
AttachTime          : 12/22/2015 1:53:58 AM
DeleteOnTermination : False
Device              : /dev/sdh
```

```
InstanceId          : i-1a2b3c4d
State               : detaching
VolumeId            : vol-12345678
```

Beispiel 2: Sie können auch die Instanz-ID und den Gerätenamen angeben, um sicherzustellen, dass Sie das richtige Volume trennen.

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

- Einzelheiten zur API finden Sie unter DetachVolume AWS Tools for PowerShellCmdlet-Referenz.

`Dismount-EC2VpnGateway`

Das folgende Codebeispiel zeigt die Verwendung. `Dismount-EC2VpnGateway`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Virtual Private Gateway von der angegebenen VPC getrennt.

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

- Einzelheiten zur API finden Sie unter DetachVpnGateway AWS Tools for PowerShellCmdlet-Referenz.

`Edit-EC2CapacityReservation`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-EC2CapacityReservation`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird CapacityReservationId cr-0c1f2345db6f7cdba geändert, indem die Anzahl der Instanzen auf 1 geändert wird

```
Edit-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba -
InstanceCount 1
```

Ausgabe:

```
True
```

- Einzelheiten AWS Tools for PowerShell zur ModifyCapacityReservationAPI finden Sie unter Cmdlet-Referenz.

`Edit-EC2Host`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-EC2Host`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die AutoPlacement Einstellungen für den dedizierten Host h-01e23f4cd567890f3 auf Aus geändert

```
Edit-EC2Host -HostId h-03e09f8cd681609f3 -AutoPlacement off
```

Ausgabe:

```
Successful           Unsuccessful
----------           ------------
{h-01e23f4cd567890f3} {}
```

- Einzelheiten AWS Tools for PowerShell zur ModifyHostsAPI finden Sie unter Cmdlet-Referenz.

`Edit-EC2IdFormat`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-EC2IdFormat`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das längere ID-Format für den angegebenen Ressourcentyp aktiviert.

```
Edit-EC2IdFormat -Resource instance -UseLongId $true
```

Beispiel 2: In diesem Beispiel wird das längere ID-Format für den angegebenen Ressourcentyp deaktiviert.

```
Edit-EC2IdFormat -Resource instance -UseLongId $false
```

- Einzelheiten zur API finden Sie unter ModifyIdFormat AWS Tools for PowerShellCmdlet-Referenz.

`Edit-EC2ImageAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-EC2ImageAttribute`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Beschreibung für das angegebene AMI aktualisiert.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Description "New description"
```

Beispiel 2: In diesem Beispiel wird das AMI öffentlich gemacht (damit es beispielsweise von jedem verwendet AWS-Konto werden kann).

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType add -UserGroup all
```

Beispiel 3: In diesem Beispiel wird das AMI privat (zum Beispiel, sodass nur Sie als Besitzer es verwenden können).

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType remove -UserGroup all
```

Beispiel 4: In diesem Beispiel wird dem angegebenen Benutzer die Startberechtigung erteilt AWS-Konto.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType add -UserId 111122223333
```

Beispiel 5: In diesem Beispiel wird die Startberechtigung für das angegebene Objekt entfernt AWS-Konto.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType remove -UserId 111122223333
```

- Einzelheiten zur API finden Sie unter ModifyImageAttribute AWS Tools for PowerShellCmdlet-Referenz.

`Edit-EC2InstanceAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-EC2InstanceAttribute`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der Instanztyp der angegebenen Instanz geändert.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceType m3.medium
```

Beispiel 2: In diesem Beispiel wird Enhanced Networking für die angegebene Instance aktiviert, indem „simple“ als Wert für den Netzwerkunterstützungsparameter Single Root I/O Virtualization (SR-IOV) angegeben wird, -.. SriovNetSupport

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SriovNetSupport "simple"
```

Beispiel 3: In diesem Beispiel werden die Sicherheitsgruppen für die angegebene Instanz geändert. Die Instance muss sich in einer VPC befinden. Sie müssen die ID jeder Sicherheitsgruppe angeben, nicht den Namen.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -Group @( "sg-12345678",
 "sg-45678901" )
```

Beispiel 4: Dieses Beispiel aktiviert die EBS-I/O-Optimierung für die angegebene Instance. Diese Funktion ist nicht für alle Instance-Typen verfügbar. Bei Verwendung einer EBS-optimierten Instance fallen zusätzliche Nutzungsgebühren an.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -EbsOptimized $true
```

Beispiel 5: In diesem Beispiel wird die Quell-/Zielüberprüfung für die angegebene Instance aktiviert. Damit eine NAT-Instance NAT ausführen kann, muss der Wert „false“ sein.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SourceDestCheck $true
```

Beispiel 6: In diesem Beispiel wird die Kündigung für die angegebene Instance deaktiviert.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -DisableApiTermination $true
```

Beispiel 7: In diesem Beispiel wird die angegebene Instanz so geändert, dass sie beendet wird, wenn das Herunterfahren von der Instance aus initiiert wird.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceInitiatedShutdownBehavior
 terminate
```

- Einzelheiten zur API finden Sie unter ModifyInstanceAttribute AWS Tools for PowerShellCmdlet-Referenz.

`Edit-EC2InstanceCreditSpecification`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-EC2InstanceCreditSpecification`

Tools für PowerShell

Beispiel 1: Dies ermöglicht unbegrenzte T2-Credits, zum Beispiel i-01234567890abcdef.

```
$Credit = New-Object -TypeName Amazon.EC2.Model.InstanceCreditSpecificationRequest
$Credit.InstanceId = "i-01234567890abcdef"
$Credit.CpuCredits = "unlimited"
Edit-EC2InstanceCreditSpecification -InstanceCreditSpecification $Credit
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz. ModifyInstanceCreditSpecificationAWS Tools for PowerShell

`Edit-EC2NetworkInterfaceAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-EC2NetworkInterfaceAttribute`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerkschnittstelle so geändert, dass die angegebene Anlage beim Beenden gelöscht wird.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -
Attachment_AttachmentId eni-attach-1a2b3c4d -Attachment_DeleteOnTermination $true
```

Beispiel 2: In diesem Beispiel wird die Beschreibung der angegebenen Netzwerkschnittstelle geändert.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Description "my
 description"
```

Beispiel 3: In diesem Beispiel wird die Sicherheitsgruppe für die angegebene Netzwerkschnittstelle geändert.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Groups
 sg-1a2b3c4d
```

Beispiel 4: In diesem Beispiel wird die Quell-/Zielüberprüfung für die angegebene Netzwerkschnittstelle deaktiviert.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -SourceDestCheck
 $false
```

- Einzelheiten zur API finden Sie unter ModifyNetworkInterfaceAttributeCmdlet-Referenz.AWS Tools for PowerShell

`Edit-EC2ReservedInstance`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-EC2ReservedInstance`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Availability Zone, die Anzahl der Instanzen und die Plattform für die angegebenen Reserved Instances geändert.

```
$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
```

- Einzelheiten zur API finden Sie unter ModifyReservedInstances AWS Tools for PowerShellCmdlet-Referenz.

`Edit-EC2SnapshotAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-EC2SnapshotAttribute`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Snapshot veröffentlicht, indem es sein CreateVolumePermission Attribut festlegt.

```
Edit-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission -OperationType Add -GroupName all
```

- Einzelheiten zur API finden Sie unter ModifySnapshotAttribute AWS Tools for PowerShellCmdlet-Referenz.

`Edit-EC2SpotFleetRequest`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-EC2SpotFleetRequest`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Zielkapazität der angegebenen Spot-Flottenanforderung aktualisiert.

```
Edit-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TargetCapacity 10
```

Ausgabe:

```
True
```

- Einzelheiten zur API finden Sie unter ModifySpotFleetRequest AWS Tools for PowerShellCmdlet-Referenz.

`Edit-EC2SubnetAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-EC2SubnetAttribute`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die öffentliche IP-Adressierung für das angegebene Subnetz aktiviert.

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $true
```

Beispiel 2: In diesem Beispiel wird die öffentliche IP-Adressierung für das angegebene Subnetz deaktiviert.

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $false
```

- Einzelheiten zur API finden Sie unter ModifySubnetAttribute AWS Tools for PowerShellCmdlet-Referenz.

`Edit-EC2VolumeAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-EC2VolumeAttribute`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Attribut des angegebenen Volumes geändert. I/O-Operationen für das Volume werden automatisch wieder aufgenommen, nachdem sie aufgrund potenziell inkonsistenter Daten unterbrochen wurden.

```
Edit-EC2VolumeAttribute -VolumeId vol-12345678 -AutoEnableIO $true
```

- Einzelheiten zur API finden Sie unter ModifyVolumeAttribute AWS Tools for PowerShellCmdlet-Referenz.

`Edit-EC2VpcAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-EC2VpcAttribute`

Tools für PowerShell

Beispiel 1: Dieses Beispiel aktiviert die Unterstützung von DNS-Hostnamen für die angegebene VPC.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $true
```

Beispiel 2: In diesem Beispiel wird die Unterstützung für DNS-Hostnamen für die angegebene VPC deaktiviert.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $false
```

Beispiel 3: In diesem Beispiel wird die Unterstützung für die DNS-Auflösung für die angegebene VPC aktiviert.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $true
```

Beispiel 4: In diesem Beispiel wird die Unterstützung für die DNS-Auflösung für die angegebene VPC deaktiviert.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $false
```

- Einzelheiten zur API finden Sie unter ModifyVpcAttribute AWS Tools for PowerShellCmdlet-Referenz.

`Enable-EC2VgwRoutePropagation`

Das folgende Codebeispiel zeigt die Verwendung. `Enable-EC2VgwRoutePropagation`

Tools für PowerShell

Beispiel 1: In diesem Beispiel kann das angegebene VGW Routen automatisch an die angegebene Routingtabelle weitergeben.

```
Enable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter EnableVgwRoutePropagationCmdlet-Referenz.AWS Tools for PowerShell

`Enable-EC2VolumeIO`

Das folgende Codebeispiel zeigt die Verwendung. `Enable-EC2VolumeIO`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden I/O-Operationen für das angegebene Volume aktiviert, wenn I/O-Operationen deaktiviert wurden.

```
Enable-EC2VolumeIO -VolumeId vol-12345678
```

- Einzelheiten zur API finden Sie unter EnableVolumeIo AWS Tools for PowerShellCmdlet-Referenz.

`Enable-EC2VpcClassicLink`

Das folgende Codebeispiel zeigt die Verwendung. `Enable-EC2VpcClassicLink`

Tools für PowerShell

Beispiel 1: Dieses Beispiel aktiviert VPC vpc-0123456b789b0d12f für ClassicLink

```
Enable-EC2VpcClassicLink -VpcId vpc-0123456b789b0d12f
```

Ausgabe:

```
True
```

- Einzelheiten zur API EnableVpcClassicLink AWS Tools for PowerShellfinden Sie unter Cmdlet-Referenz.

`Enable-EC2VpcClassicLinkDnsSupport`

Das folgende Codebeispiel zeigt die Verwendung. `Enable-EC2VpcClassicLinkDnsSupport`

Tools für PowerShell

Beispiel 1: Dieses Beispiel ermöglicht vpc-0b12d3456a7e8910d die Unterstützung der DNS-Hostnamenauflösung für ClassicLink

```
Enable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

- Einzelheiten zur EnableVpcClassicLinkDnsSupportAPI AWS Tools for PowerShell finden Sie unter Cmdlet-Referenz.

`Get-EC2AccountAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2AccountAttribute`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird beschrieben, ob Sie Instances in EC2-Classic und EC2-VPC in der Region oder nur in EC2-VPC VPC können.

```
(Get-EC2AccountAttribute -AttributeName supported-platforms).AttributeValues
```

Ausgabe:

```
AttributeValue
--------------
EC2
VPC
```

Beispiel 2: Dieses Beispiel beschreibt Ihre Standard-VPC oder ist „Keine“, wenn Sie keine Standard-VPC in der Region haben.

```
(Get-EC2AccountAttribute -AttributeName default-vpc).AttributeValues
```

Ausgabe:

```
AttributeValue
--------------
vpc-12345678
```

Beispiel 3: Dieses Beispiel beschreibt die maximale Anzahl von On-Demand-Instances, die Sie ausführen können.

```
(Get-EC2AccountAttribute -AttributeName max-instances).AttributeValues
```

Ausgabe:

```
AttributeValue
--------------
20
```

- Einzelheiten zur API finden Sie unter DescribeAccountAttributes AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2Address`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2Address`

Beispiel 1: Dieses Beispiel beschreibt die angegebene Elastic IP-Adresse für Instances in EC2-Classic.

```
Get-EC2Address -AllocationId eipalloc-12345678
```

Ausgabe:

```
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
```

Beispiel 2: Dieses Beispiel beschreibt Ihre Elastic IP-Adressen für Instances in einer VPC. Diese Syntax erfordert PowerShell Version 3 oder höher.

```
Get-EC2Address -Filter @{ Name="domain";Values="vpc" }
```

Beispiel 3: Dieses Beispiel beschreibt die angegebene Elastic IP-Adresse für Instances in EC2-Classic.

```
Get-EC2Address -PublicIp 203.0.113.17
```

Ausgabe:

```
AllocationId            :
AssociationId           :
Domain                  : standard
InstanceId              : i-12345678
NetworkInterfaceId      :
NetworkInterfaceOwnerId :
PrivateIpAddress        :
PublicIp                : 203.0.113.17
```

Beispiel 4: Dieses Beispiel beschreibt Ihre Elastic IP-Adressen für Instances in EC2-Classic. Für diese Syntax ist PowerShell Version 3 oder höher erforderlich.

```
Get-EC2Address -Filter @{ Name="domain";Values="standard" }
```

Beispiel 5: Dieses Beispiel beschreibt all Ihre Elastic IP-Adressen.

```
Get-EC2Address
```

Beispiel 6: Dieses Beispiel gibt die öffentliche und private IP für die im Filter angegebene Instance-ID zurück

```
Get-EC2Address -Region eu-west-1 -Filter @{Name="instance-
id";Values="i-0c12d3f4f567ffb89"} | Select-Object PrivateIpAddress, PublicIp
```

Ausgabe:

```
PrivateIpAddress PublicIp
---------------- --------
10.0.0.99        63.36.5.227
```

Beispiel 7: In diesem Beispiel werden alle Elastic IPs mit ihrer Zuweisungs-ID, Zuordnungs-ID und Instanz-IDs abgerufen

```
Get-EC2Address -Region eu-west-1 | Select-Object InstanceId, AssociationId,
 AllocationId, PublicIp
```

Ausgabe:

```
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
```

Beispiel 8: In diesem Beispiel wird eine Liste von EC2-IP-Adressen abgerufen, die dem Tag-Schlüssel 'Category' mit dem Wert 'Prod' entsprechen

```
Get-EC2Address -Filter @{Name="tag:Category";Values="Prod"}
```

Ausgabe:

```
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}
```

- Einzelheiten zur API finden Sie unter DescribeAddressesCmdlet-Referenz.AWS Tools for PowerShell

`Get-EC2AvailabilityZone`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2AvailabilityZone`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Availability Zones für die aktuelle Region beschrieben, die Ihnen zur Verfügung stehen.

```
Get-EC2AvailabilityZone
```

Ausgabe:

```
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
```

Beispiel 2: In diesem Beispiel werden alle Availability Zones beschrieben, die sich in einem beeinträchtigten Zustand befinden. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Get-EC2AvailabilityZone -Filter @{ Name="state";Values="impaired" }
```

Beispiel 3: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um den Filter zu erstellen.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = "impaired"

Get-EC2AvailabilityZone -Filter $filter
```

- Einzelheiten zur API finden Sie unter DescribeAvailabilityZones AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2BundleTask`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2BundleTask`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die angegebene Bundle-Aufgabe.

```
Get-EC2BundleTask -BundleId bun-12345678
```

Beispiel 2: Dieses Beispiel beschreibt die Bundle-Aufgaben, deren Status entweder „abgeschlossen“ oder „fehlgeschlagen“ ist.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "complete", "failed" )
```

```
Get-EC2BundleTask -Filter $filter
```

- Einzelheiten zur API finden Sie unter DescribeBundleTasksCmdlet-Referenz.AWS Tools for PowerShell

`Get-EC2CapacityReservation`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2CapacityReservation`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt eine oder mehrere Ihrer Kapazitätsreservierungen für die Region

```
Get-EC2CapacityReservation -Region eu-west-1
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeCapacityReservations AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2ConsoleOutput`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2ConsoleOutput`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Konsolenausgabe für die angegebene Linux-Instance abgerufen. Die Konsolenausgabe ist codiert.

```
Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456
```

Ausgabe:

```
InstanceId          Output
----------          ------
i-0e194d3c47c123637 WyAgICAwLjAwMDAwMF0gQ29tbW...bGU9dHR5UzAgc2Vs
```

Beispiel 2: In diesem Beispiel wird die kodierte Konsolenausgabe in einer Variablen gespeichert und anschließend dekodiert.

```
$Output_encoded = (Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456).Output
[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($Output_encoded))
```

- Einzelheiten zur API finden Sie unter GetConsoleOutput AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2CustomerGateway`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2CustomerGateway`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt das angegebene Kunden-Gateway.

```
Get-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

Ausgabe:

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

Beispiel 2: Dieses Beispiel beschreibt jedes Kunden-Gateway, dessen Status entweder ausstehend oder verfügbar ist.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2CustomerGateway -Filter $filter
```

Beispiel 3: Dieses Beispiel beschreibt alle Ihre Kunden-Gateways.

```
Get-EC2CustomerGateway
```

- Einzelheiten zur API finden Sie unter DescribeCustomerGateways AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2DhcpOption`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2DhcpOption`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Ihre DHCP-Optionssätze aufgeführt.

```
Get-EC2DhcpOption
```

Ausgabe:

```
DhcpConfigurations                    DhcpOptionsId    Tag
------------------                    -------------    ---
{domain-name, domain-name-servers}    dopt-1a2b3c4d    {}
{domain-name, domain-name-servers}    dopt-2a3b4c5d    {}
{domain-name-servers}                 dopt-3a4b5c6d    {}
```

Beispiel 2: In diesem Beispiel werden Konfigurationsdetails für den angegebenen DHCP-Optionssatz abgerufen.

```
(Get-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d).DhcpConfigurations
```

Ausgabe:

```
Key                   Values
---                   ------
domain-name           {abc.local}
domain-name-servers   {10.0.0.101, 10.0.0.102}
```

- Einzelheiten zur API finden Sie unter DescribeDhcpOptions AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2FlowLog`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2FlowLog`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt ein oder mehrere Flow-Logs mit dem Protokollzieltyp 's3'

```
Get-EC2FlowLog -Filter @{Name="log-destination-type";Values="s3"}
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeFlowLogs AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2Host`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2Host`

Beispiel 1: In diesem Beispiel werden die EC2-Host-Details zurückgegeben

```
Get-EC2Host
```

Ausgabe:

```
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              : {}
```

Beispiel 2: In diesem Beispiel wird nach dem Host AvailableInstanceCapacity h-01e23f4cd567899f1 abgefragt

```
Get-EC2Host -HostId h-01e23f4cd567899f1 | Select-Object -ExpandProperty
 AvailableCapacity | Select-Object -expand AvailableInstanceCapacity
```

Ausgabe:

```
AvailableCapacity InstanceType TotalCapacity
----------------- ------------ -------------
11                m4.xlarge    11
```

- Einzelheiten AWS Tools for PowerShell zur DescribeHostsAPI finden Sie unter Cmdlet-Referenz.

`Get-EC2HostReservationOffering`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2HostReservationOffering`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Dedicated Host-Reservierungen beschrieben, die für den angegebenen Filter „Instance-Familie“ erworben werden können, wobei der Filter „“ PaymentOption lautet NoUpfront

```
Get-EC2HostReservationOffering -Filter @{Name="instance-family";Values="m4"} |
 Where-Object PaymentOption -eq NoUpfront
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeHostReservationOfferings AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2HostReservationPurchasePreview`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2HostReservationPurchasePreview`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Vorschau eines Reservierungskaufs mit Konfigurationen angezeigt, die denen Ihres Dedicated Hosts h-01e23f4cd567890f1 entsprechen

```
Get-EC2HostReservationPurchasePreview -OfferingId hro-0c1f23456789d0ab -HostIdSet
 h-01e23f4cd567890f1
```

Ausgabe:

```
CurrencyCode Purchase TotalHourlyPrice TotalUpfrontPrice
------------ -------- ---------------- -----------------
             {}       1.307            0.000
```

- Einzelheiten GetHostReservationPurchasePreviewzur API AWS Tools for PowerShell finden Sie unter Cmdlet-Referenz.

`Get-EC2IdFormat`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2IdFormat`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt das ID-Format für den angegebenen Ressourcentyp.

```
Get-EC2IdFormat -Resource instance
```

Ausgabe:

```
Resource       UseLongIds
--------       ----------
instance       False
```

Beispiel 2: Dieses Beispiel beschreibt die ID-Formate für alle Ressourcentypen, die längere IDs unterstützen.

```
Get-EC2IdFormat
```

Ausgabe:

```
Resource       UseLongIds
--------       ----------
reservation    False
instance       False
```

- Einzelheiten zur API finden Sie unter DescribeIdFormat AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2IdentityIdFormat`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2IdentityIdFormat`

Tools für PowerShell

Beispiel 1: Dieses Beispiel gibt das ID-Format für die Ressource 'image' für die angegebene Rolle zurück

```
Get-EC2IdentityIdFormat -PrincipalArn arn:aws:iam::123456789511:role/JDBC -Resource
 image
```

Ausgabe:

```
Deadline             Resource UseLongIds
--------             -------- ----------
8/2/2018 11:30:00 PM image    True
```

- Einzelheiten zur API finden Sie unter DescribeIdentityIdFormat AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2Image`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2Image`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt das angegebene AMI.

```
Get-EC2Image -ImageId ami-12345678
```

Ausgabe:

```
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
```

Beispiel 2: Dieses Beispiel beschreibt die AMIs, die Sie besitzen.

```
Get-EC2Image -owner self
```

Beispiel 3: In diesem Beispiel werden die öffentlichen AMIs beschrieben, auf denen Microsoft Windows Server ausgeführt wird.

```
Get-EC2Image -Filter @{ Name="platform"; Values="windows" }
```

Beispiel 4: Dieses Beispiel beschreibt alle öffentlichen AMIs in der Region „us-west-2“.

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

- Einzelheiten zur API finden Sie unter DescribeImages AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2ImageAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2ImageAttribute`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Beschreibung für das angegebene AMI abgerufen.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute description
```

Ausgabe:

```
BlockDeviceMappings : {}
Description         : My image description
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {}
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     :
```

Beispiel 2: In diesem Beispiel werden die Startberechtigungen für das angegebene AMI abgerufen.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

Ausgabe:

```
BlockDeviceMappings : {}
Description         :
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {all}
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     :
```

Beispiel 3: In diesem Beispiel wird getestet, ob Enhanced Networking aktiviert ist.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute sriovNetSupport
```

Ausgabe:

```
BlockDeviceMappings : {}
Description         :
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {}
ProductCodes        : {}
```

```
RamdiskId           :
SriovNetSupport     : simple
```

- Einzelheiten zur API finden Sie unter DescribeImageAttribute AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2ImageByName`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2ImageByName`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt den vollständigen Satz von Filternamen, die derzeit unterstützt werden.

```
Get-EC2ImageByName
```

Ausgabe:

```
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
```

Beispiel 2: Dieses Beispiel beschreibt das angegebene AMI. Die Verwendung dieses Befehls zur Suche nach einem AMI ist hilfreich, da jeden Monat neue Windows-AMIs mit den neuesten Updates AWS veröffentlicht werden. Sie können das 'ImageId' angeben, New-EC2Instance um eine Instance mit dem aktuellen AMI für den angegebenen Filter zu starten.

```
Get-EC2ImageByName -Names WINDOWS_2016_BASE
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter Get-EC2ImageByName AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2ImportImageTask`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2ImportImageTask`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die angegebene Bildimportaufgabe.

```
Get-EC2ImportImageTask -ImportTaskId import-ami-hgfedcba
```

Ausgabe:

```
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   :
```

Beispiel 2: In diesem Beispiel werden alle Ihre Bildimportaufgaben beschrieben.

```
Get-EC2ImportImageTask
```

Ausgabe:

```
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   :
```

- Einzelheiten zur API finden Sie unter DescribeImportImageTasks AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2ImportSnapshotTask`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2ImportSnapshotTask`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die angegebene Snapshot-Importaufgabe.

```
Get-EC2ImportSnapshotTask -ImportTaskId import-snap-abcdefgh
```

Ausgabe:

```
Description             ImportTaskId              SnapshotTaskDetail
```

```
-----------------       --------------------       ------------------

Disk Image Import 1     import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
```

Beispiel 2: In diesem Beispiel werden alle Ihre Snapshot-Importaufgaben beschrieben.

```
Get-EC2ImportSnapshotTask
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeImportSnapshotTasks AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2Instance`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2Instance`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die angegebene Instanz.

```
(Get-EC2Instance -InstanceId i-12345678).Instances
```

Ausgabe:

```
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
```

Beispiel 2: Dieses Beispiel beschreibt alle Ihre Instances in der aktuellen Region, gruppiert nach Reservierungen. Um die Instance-Details zu sehen, erweitern Sie die Instance-Sammlung innerhalb jedes Reservierungsobjekts.

```
Get-EC2Instance
```

Ausgabe:

```
GroupNames    : {}
```

```
Groups        : {}
Instances     : {}
OwnerId       : 123456789012
RequesterId   : 226008221399
ReservationId : r-c5df370c

GroupNames    : {}
Groups        : {}
Instances     : {}
OwnerId       : 123456789012
RequesterId   : 854251627541
ReservationId : r-63e65bab
...
```

Beispiel 3: Dieses Beispiel veranschaulicht die Verwendung eines Filters zur Abfrage von EC2-Instances in einem bestimmten Subnetz einer VPC.

```
(Get-EC2Instance -Filter @{Name="vpc-id";Values="vpc-1a2bc34d"},@{Name="subnet-
id";Values="subnet-1a2b3c4d"}).Instances
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeInstancesCmdlet-Referenz.AWS Tools for PowerShell

`Get-EC2InstanceAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2InstanceAttribute`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt den Instanztyp der angegebenen Instanz.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute instanceType
```

Ausgabe:

```
InstanceType                     : t2.micro
```

Beispiel 2: In diesem Beispiel wird beschrieben, ob Enhanced Networking für die angegebene Instanz aktiviert ist.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

Ausgabe:

```
SriovNetSupport                  : simple
```

Beispiel 3: In diesem Beispiel werden die Sicherheitsgruppen für die angegebene Instance beschrieben.

```
(Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute groupSet).Groups
```

Ausgabe:

```
GroupId
-------
sg-12345678
sg-45678901
```

Beispiel 4: In diesem Beispiel wird beschrieben, ob die EBS-Optimierung für die angegebene Instance aktiviert ist.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

Ausgabe:

```
EbsOptimized                     : False
```

Beispiel 5: Dieses Beispiel beschreibt das Attribut 'disableApiTermination' der angegebenen Instance.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

Ausgabe:

```
DisableApiTermination            : False
```

Beispiel 6: Dieses Beispiel beschreibt das Attribut 'instanceInitiatedShutdownBehavior' der angegebenen Instanz.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
 instanceInitiatedShutdownBehavior
```

Ausgabe:

```
InstanceInitiatedShutdownBehavior : stop
```

- Einzelheiten zur API finden Sie unter DescribeInstanceAttribute AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2InstanceMetadata`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2InstanceMetadata`

Tools für PowerShell

Beispiel 1: Listet die verfügbaren Kategorien von Instanz-Metadaten auf, die abgefragt werden können.

```
Get-EC2InstanceMetadata -ListCategory
```

Ausgabe:

```
AmiId
LaunchIndex
ManifestPath
AncestorAmiId
BlockDeviceMapping
InstanceId
```

```
InstanceType
LocalHostname
LocalIpv4
KernelId
AvailabilityZone
ProductCode
PublicHostname
PublicIpv4
PublicKey
RamdiskId
Region
ReservationId
SecurityGroup
UserData
InstanceMonitoring
IdentityDocument
IdentitySignature
IdentityPkcs7
```

Beispiel 2: Gibt die ID des Amazon Machine Image (AMI) zurück, das zum Starten der Instance verwendet wurde.

```
Get-EC2InstanceMetadata -Category AmiId
```

Ausgabe:

```
ami-b2e756ca
```

Beispiel 3: In diesem Beispiel wird das Ausweisdokument im JSON-Format für die Instance abgefragt.

```
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"
}
```

Beispiel 4: In diesem Beispiel wird eine Pfadabfrage verwendet, um die Netzwerkschnittstellen-Macs für die Instance abzurufen.

```
Get-EC2InstanceMetadata -Path "/network/interfaces/macs"
```

Ausgabe:

```
02:80:7f:ef:4c:e0/
```

Beispiel 5: Wenn der Instance eine IAM-Rolle zugeordnet ist, werden Informationen darüber zurückgegeben, wann das Instanzprofil zuletzt aktualisiert wurde, einschließlich des LastUpdated Datums der Instanz InstanceProfileArn, und. InstanceProfileId

```
Get-EC2InstanceMetadata -Path "/iam/info"
```

Ausgabe:

```
{
  "Code" : "Success",
  "LastUpdated" : "2018-03-08T03:38:40Z",
  "InstanceProfileArn" : "arn:aws:iam::111122223333:instance-profile/
MyLaunchRole_Profile",
  "InstanceProfileId" : "AIPAI4...WVK2RW"
}
```

- Einzelheiten zur API finden Sie unter Get-EC2InstanceMetadata AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2InstanceStatus`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2InstanceStatus`

Beispiel 1: Dieses Beispiel beschreibt den Status der angegebenen Instanz.

```
Get-EC2InstanceStatus -InstanceId i-12345678
```

Ausgabe:

```
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
```

Ausgabe:

```
Code    Name
----    ----
16      running
```

```
$status.Status
```

Ausgabe:

```
Details           Status
-------           ------
{reachability}    ok
```

```
$status.SystemStatus
```

Ausgabe:

```
Details           Status
-------           ------
{reachability}    ok
```

- Einzelheiten zur API finden Sie unter DescribeInstanceStatus AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2InternetGateway`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2InternetGateway`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt das angegebene Internet-Gateway.

```
Get-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

Ausgabe:

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

Beispiel 2: Dieses Beispiel beschreibt alle Ihre Internet-Gateways.

```
Get-EC2InternetGateway
```

Ausgabe:

```
Attachments       InternetGatewayId     Tags
-----------       -----------------     ----
{vpc-1a2b3c4d}    igw-1a2b3c4d          {}
{}                igw-2a3b4c5d          {}
```

- Einzelheiten zur API finden Sie unter DescribeInternetGateways AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2KeyPair`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2KeyPair`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt das angegebene key pair.

```
Get-EC2KeyPair -KeyName my-key-pair
```

Ausgabe:

```
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
```

Beispiel 2: Dieses Beispiel beschreibt alle Ihre Schlüsselpaare.

```
Get-EC2KeyPair
```

- Einzelheiten zur API finden Sie unter DescribeKeyPairs AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2NetworkAcl`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2NetworkAcl`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die angegebene Netzwerk-ACL.

```
Get-EC2NetworkAcl -NetworkAclId acl-12345678
```

Ausgabe:

```
Associations : {aclassoc-1a2b3c4d}
Entries      : {Amazon.EC2.Model.NetworkAclEntry, Amazon.EC2.Model.NetworkAclEntry}
IsDefault    : False
NetworkAclId : acl-12345678
Tags         : {Name}
VpcId        : vpc-12345678
```

Beispiel 2: Dieses Beispiel beschreibt die Regeln für die angegebene Netzwerk-ACL.

```
(Get-EC2NetworkAcl -NetworkAclId acl-12345678).Entries
```

Ausgabe:

```
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
```

Beispiel 3: Dieses Beispiel beschreibt alle Ihre Netzwerk-ACLs.

```
Get-EC2NetworkAcl
```

- Einzelheiten zur API finden Sie unter DescribeNetworkAcls AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2NetworkInterface`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2NetworkInterface`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die angegebene Netzwerkschnittstelle.

```
Get-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

Ausgabe:

```
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
```

Beispiel 2: Dieses Beispiel beschreibt alle Ihre Netzwerkschnittstellen.

```
Get-EC2NetworkInterface
```

- Einzelheiten zur API finden Sie unter DescribeNetworkInterfaces AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2NetworkInterfaceAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2NetworkInterfaceAttribute`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die angegebene Netzwerkschnittstelle.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 Attachment
```

Ausgabe:

```
Attachment         : Amazon.EC2.Model.NetworkInterfaceAttachment
```

Beispiel 2: Dieses Beispiel beschreibt die angegebene Netzwerkschnittstelle.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 Description
```

Ausgabe:

```
Description        : My description
```

Beispiel 3: Dieses Beispiel beschreibt die angegebene Netzwerkschnittstelle.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 GroupSet
```

Ausgabe:

```
Groups             : {my-security-group}
```

Beispiel 4: Dieses Beispiel beschreibt die angegebene Netzwerkschnittstelle.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 SourceDestCheck
```

Ausgabe:

```
SourceDestCheck    : True
```

- Einzelheiten zur API finden Sie unter DescribeNetworkInterfaceAttribute AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2PasswordData`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2PasswordData`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Passwort entschlüsselt, das Amazon EC2 dem Administratorkonto für die angegebene Windows-Instance zugewiesen hat. Da eine PEM-Datei angegeben wurde, wird automatisch die Einstellung des Schalters -Drypt übernommen.

```
Get-EC2PasswordData -InstanceId i-12345678 -PemFile C:\path\my-key-pair.pem
```

Ausgabe:

```
mYZ(PA9?C)Q
```

Beispiel 2: (PowerShell nur Windows) Überprüft die Instanz, um den Namen des Schlüsselpaars zu ermitteln, das zum Starten der Instanz verwendet wurde, und versucht dann, die entsprechenden Schlüsselpaardaten im Konfigurationsspeicher des AWS Toolkit for Visual Studio zu finden. Wenn die Schlüsselpaardaten gefunden werden, wird das Passwort entschlüsselt.

```
Get-EC2PasswordData -InstanceId i-12345678 -Decrypt
```

Ausgabe:

```
mYZ(PA9?C)Q
```

Beispiel 3: Gibt die verschlüsselten Passwortdaten für die Instanz zurück.

```
Get-EC2PasswordData -InstanceId i-12345678
```

Ausgabe:

```
iVz3BAK/WAXV.....dqt8WeMA==
```

- Einzelheiten zur API finden Sie unter GetPasswordData AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2PlacementGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2PlacementGroup`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die angegebene Platzierungsgruppe.

```
Get-EC2PlacementGroup -GroupName my-placement-group
```

Ausgabe:

```
GroupName            State       Strategy
---------            -----       --------
```

```
my-placement-group    available    cluster
```

- Einzelheiten zur API finden Sie unter DescribePlacementGroups AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2PrefixList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2PrefixList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das AWS-Services in einer Präfixliste verfügbare Format für die Region abgerufen

```
Get-EC2PrefixList
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribePrefixLists AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2Region`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2Region`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Regionen beschrieben, die Ihnen zur Verfügung stehen.

```
Get-EC2Region
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeRegions AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2RouteTable`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2RouteTable`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt alle Ihre Routentabellen.

```
Get-EC2RouteTable
```

Ausgabe:

```
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  :
```

Beispiel 2: In diesem Beispiel werden Details für die angegebene Routentabelle zurückgegeben.

```
Get-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

Beispiel 3: Dieses Beispiel beschreibt die Routentabellen für die angegebene VPC.

```
Get-EC2RouteTable -Filter @{ Name="vpc-id"; Values="vpc-1a2b3c4d" }
```

Ausgabe:

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

- Einzelheiten zur API finden Sie unter DescribeRouteTables AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2ScheduledInstance`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2ScheduledInstance`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die angegebene geplante Instanz.

```
Get-EC2ScheduledInstance -ScheduledInstanceId sci-1234-1234-1234-1234-123456789012
```

Ausgabe:

```
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
```

Beispiel 2: Dieses Beispiel beschreibt all Ihre geplanten Instances.

```
Get-EC2ScheduledInstance
```

- Einzelheiten zur API finden Sie unter DescribeScheduledInstances AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2ScheduledInstanceAvailability`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2ScheduledInstanceAvailability`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt einen Zeitplan, der jede Woche am Sonntag beginnt und am angegebenen Datum beginnt.

```
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
```

Ausgabe:

```
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

...
```

Beispiel 2: Um die Ergebnisse einzugrenzen, können Sie Filter für Kriterien wie Betriebssystem, Netzwerk und Instanztyp hinzufügen.

```
-Filter @{ Name="platform";Values="Linux/UNIX" },@{ Name="network-
platform";Values="EC2-VPC" },@{ Name="instance-type";Values="c4.large" }
```

- Einzelheiten zur API finden Sie unter DescribeScheduledInstanceAvailability AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2SecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2SecurityGroup`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die angegebene Sicherheitsgruppe für eine VPC. Wenn Sie mit Sicherheitsgruppen arbeiten, die zu einer VPC gehören, müssen Sie die Sicherheitsgruppen-ID (- GroupId Parameter) und nicht den Namen (- GroupName Parameter) verwenden, um auf die Gruppe zu verweisen.

```
Get-EC2SecurityGroup -GroupId sg-12345678
```

Ausgabe:

```
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
```

Beispiel 2: Dieses Beispiel beschreibt die angegebene Sicherheitsgruppe für EC2-Classic. Wenn Sie mit Sicherheitsgruppen für EC2-Classic arbeiten, können Sie entweder den Gruppennamen (- GroupName Parameter) oder die Gruppen-ID (- GroupId Parameter) verwenden, um auf die Sicherheitsgruppe zu verweisen.

```
Get-EC2SecurityGroup -GroupName my-security-group
```

Ausgabe:

```
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               :
```

Beispiel 3: In diesem Beispiel werden alle Sicherheitsgruppen für die Datei vpc-0fc1ff23456b789eb abgerufen

```
Get-EC2SecurityGroup -Filter @{Name="vpc-id";Values="vpc-0fc1ff23456b789eb"}
```

- Einzelheiten zur DescribeSecurityGroupsAPI AWS Tools for PowerShell finden Sie unter Cmdlet-Referenz.

`Get-EC2Snapshot`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2Snapshot`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt den angegebenen Snapshot.

```
Get-EC2Snapshot -SnapshotId snap-12345678
```

Ausgabe:

```
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
```

Beispiel 2: Dieses Beispiel beschreibt die Snapshots mit dem Tag „Name“.

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" }
```

Beispiel 3: Dieses Beispiel beschreibt die Schnappschüsse, die ein 'Name' -Tag mit dem Wert '' TestValue haben.

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" -and
 $_.Tags.Value -eq "TestValue" }
```

Beispiel 4: Dieses Beispiel beschreibt all Ihre Schnappschüsse.

```
Get-EC2Snapshot -Owner self
```

- Einzelheiten zur API finden Sie unter DescribeSnapshots AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2SnapshotAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2SnapshotAttribute`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt das angegebene Attribut des angegebenen Snapshots.

```
Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute ProductCodes
```

Ausgabe:

```
CreateVolumePermissions    ProductCodes    SnapshotId
-----------------------    ------------    ----------
{}                         {}              snap-12345678
```

Beispiel 2: Dieses Beispiel beschreibt das angegebene Attribut des angegebenen Snapshots.

```
(Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission).CreateVolumePermissions
```

Ausgabe:

```
Group    UserId
-----    ------
all
```

- Einzelheiten zur API finden Sie unter DescribeSnapshotAttribute AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2SpotDatafeedSubscription`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2SpotDatafeedSubscription`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt Ihren Spot-Instance-Datenfeed.

```
Get-EC2SpotDatafeedSubscription
```

Ausgabe:

```
Bucket  : my-s3-bucket
```

```
Fault   :
OwnerId : 123456789012
Prefix  : spotdata
State   : Active
```

- Einzelheiten zur API finden Sie unter DescribeSpotDatafeedSubscription AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2SpotFleetInstance`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2SpotFleetInstance`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Instances beschrieben, die mit der angegebenen Spot-Flottenanforderung verknüpft sind.

```
Get-EC2SpotFleetInstance -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE
```

Ausgabe:

```
InstanceId    InstanceType    SpotInstanceRequestId
----------    ------------    ---------------------
i-f089262a    c3.large        sir-12345678
i-7e8b24a4    c3.large        sir-87654321
```

- Einzelheiten zur API finden Sie unter DescribeSpotFleetInstances AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2SpotFleetRequest`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2SpotFleetRequest`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die angegebene Spot-Flottenanfrage.

```
Get-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
 | format-list
```

Ausgabe:

```
ConfigData            : Amazon.EC2.Model.SpotFleetRequestConfigData
CreateTime            : 12/26/2015 8:23:33 AM
SpotFleetRequestId    : sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
SpotFleetRequestState : active
```

Beispiel 2: Dieses Beispiel beschreibt alle Ihre Spot-Flottenanfragen.

```
Get-EC2SpotFleetRequest
```

- Einzelheiten zur API finden Sie unter DescribeSpotFleetRequests AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2SpotFleetRequestHistory`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2SpotFleetRequestHistory`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt den Verlauf der angegebenen Spot-Flottenanfrage.

```
Get-EC2SpotFleetRequestHistory -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -StartTime 2015-12-26T00:00:00Z
```

Ausgabe:

```
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
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeSpotFleetRequestHistory AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2SpotInstanceRequest`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2SpotInstanceRequest`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die angegebene Spot-Instance-Anfrage.

```
Get-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

Ausgabe:

```
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
```

Beispiel 2: Dieses Beispiel beschreibt alle Ihre Spot-Instance-Anfragen.

```
Get-EC2SpotInstanceRequest
```

- Einzelheiten zur API finden Sie unter DescribeSpotInstanceRequests AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2SpotPriceHistory`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2SpotPriceHistory`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die letzten 10 Einträge in der Spot-Preishistorie für den angegebenen Instance-Typ und die Availability Zone abgerufen. Beachten Sie, dass der für den AvailabilityZone Parameter - angegebene Wert für den Regionswert gültig sein muss, der entweder an den Parameter -Region des Cmdlets übergeben wurde (im Beispiel nicht gezeigt) oder als Standard in der Shell festgelegt wurde. Bei diesem Beispielbefehl wird davon ausgegangen, dass die Standardregion 'us-west-2' in der Umgebung festgelegt wurde.

```
Get-EC2SpotPriceHistory -InstanceType c3.large -AvailabilityZone us-west-2a -
MaxResult 10
```

Ausgabe:

```
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
```

```
...
```

- Einzelheiten zur API finden Sie unter DescribeSpotPriceHistory AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2Subnet`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2Subnet`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt das angegebene Subnetz.

```
Get-EC2Subnet -SubnetId subnet-1a2b3c4d
```

Ausgabe:

```
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
```

Beispiel 2: Dieses Beispiel beschreibt alle Ihre Subnetze.

```
Get-EC2Subnet
```

- Einzelheiten zur API finden Sie unter DescribeSubnets AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2Tag`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2Tag`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Tags für den Ressourcentyp 'image' abgerufen

```
Get-EC2Tag -Filter @{Name="resource-type";Values="image"}
```

Ausgabe:

```
Key         ResourceId            ResourceType Value
---         ----------            ------------ -----
Name        ami-0a123b4ccb567a8ea image        Win7-Imported
auto-delete ami-0a123b4ccb567a8ea image        never
```

Beispiel 2: In diesem Beispiel werden alle Tags für alle Ressourcen abgerufen und nach Ressourcentyp gruppiert

```
Get-EC2Tag | Group-Object resourcetype
```

Ausgabe:

```
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}
```

Beispiel 3: In diesem Beispiel werden alle Ressourcen mit dem Tag 'auto-delete' und dem Wert 'no' für die angegebene Region angezeigt

```
Get-EC2Tag -Region eu-west-1 -Filter @{Name="tag:auto-delete";Values="no"}
```

Ausgabe:

```
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
```

Beispiel 4: In diesem Beispiel werden alle Ressourcen mit dem Tag 'auto-delete' mit dem Wert 'no' und weitere Filter in der nächsten Pipe abgerufen, um nur die Ressourcentypen 'Instanz' zu analysieren, und erstellt schließlich das Tag 'ThisInstance' für jede Instanzressource, wobei der Wert die Instanz-ID selbst ist

```
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}}
```

Beispiel 5: In diesem Beispiel werden Tags für alle Instanzressourcen sowie Namensschlüssel abgerufen und in einem Tabellenformat angezeigt

```
Get-EC2Tag -Filter @{Name="resource-
type";Values="instance"},@{Name="key";Values="Name"} | Select-Object ResourceId,
 @{Name="Name-Tag";Expression={$PSItem.Value}} | Format-Table -AutoSize
```

Ausgabe:

```
ResourceId          Name-Tag
----------          --------
i-012e3cb4df567e1aa jump1
i-01c23a45d6fc7a89f repro-3
```

- Einzelheiten zur API finden Sie unter DescribeTags AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2Volume`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2Volume`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt das angegebene EBS-Volume.

```
Get-EC2Volume -VolumeId vol-12345678
```

Ausgabe:

```
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
```

Beispiel 2: In diesem Beispiel werden Ihre EBS-Volumes beschrieben, die den Status „verfügbar“ haben.

```
Get-EC2Volume -Filter @{ Name="status"; Values="available" }
```

Ausgabe:

```
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
...
```

Beispiel 3: Dieses Beispiel beschreibt alle Ihre EBS-Volumes.

```
Get-EC2Volume
```

- Einzelheiten zur API finden Sie unter DescribeVolumes AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2VolumeAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2VolumeAttribute`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt das angegebene Attribut des angegebenen Volumes.

```
Get-EC2VolumeAttribute -VolumeId vol-12345678 -Attribute AutoEnableIO
```

Ausgabe:

```
AutoEnableIO    ProductCodes    VolumeId
------------    ------------    --------
False           {}              vol-12345678
```

- Einzelheiten zur API finden Sie unter DescribeVolumeAttribute AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2VolumeStatus`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2VolumeStatus`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt den Status des angegebenen Volumes.

```
Get-EC2VolumeStatus -VolumeId vol-12345678
```

Ausgabe:

```
Actions          : {}
AvailabilityZone : us-west-2a
Events           : {}
VolumeId         : vol-12345678
VolumeStatus     : Amazon.EC2.Model.VolumeStatusInfo
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus
```

Ausgabe:

```
Details                         Status
-------                         ------
{io-enabled, io-performance}    ok
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus.Details
```

Ausgabe:

```
Name                            Status
----                            ------
io-enabled                      passed
io-performance                  not-applicable
```

- Einzelheiten zur API finden Sie unter DescribeVolumeStatus AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2Vpc`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2Vpc`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die angegebene VPC.

```
Get-EC2Vpc -VpcId vpc-12345678
```

Ausgabe:

```
CidrBlock       : 10.0.0.0/16
DhcpOptionsId   : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault       : False
State           : available
Tags            : {Name}
VpcId           : vpc-12345678
```

Beispiel 2: Dieses Beispiel beschreibt die Standard-VPC (es kann nur eine pro Region geben). Wenn Ihr Konto EC2-Classic in dieser Region unterstützt, gibt es keine Standard-VPC.

```
Get-EC2Vpc -Filter @{Name="isDefault"; Values="true"}
```

Ausgabe:

```
CidrBlock       : 172.31.0.0/16
DhcpOptionsId   : dopt-12345678
InstanceTenancy : default
IsDefault       : True
State           : available
Tags            : {}
VpcId           : vpc-45678901
```

Beispiel 3: Dieses Beispiel beschreibt die VPCs, die dem angegebenen Filter entsprechen (d. h. über eine CIDR verfügen, die dem Wert '10.0.0.0/16' entspricht und die sich im Status 'verfügbar' befinden).

```
Get-EC2Vpc -Filter @{Name="cidr";
 Values="10.0.0.0/16"},@{Name="state";Values="available"}
```

Beispiel 4: Dieses Beispiel beschreibt alle Ihre VPCs.

```
Get-EC2Vpc
```

- Einzelheiten zur API finden Sie unter DescribeVpcs AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2VpcAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2VpcAttribute`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Attribut 'enableDnsSupport' beschrieben.

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsSupport
```

Ausgabe:

```
EnableDnsSupport
----------------
True
```

Beispiel 2: In diesem Beispiel wird das Attribut 'enableDnsHostnames' beschrieben.

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsHostnames
```

Ausgabe:

```
EnableDnsHostnames
------------------
True
```

- Einzelheiten zur API finden Sie unter DescribeVpcAttribute AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2VpcClassicLink`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2VpcClassicLink`

Tools für PowerShell

Beispiel 1: Das obige Beispiel gibt alle VPCs mit ihrem ClassicLinkEnabled Status für die Region zurück

```
Get-EC2VpcClassicLink -Region eu-west-1
```

Ausgabe:

```
ClassicLinkEnabled Tags   VpcId
------------------ ----   -----
False              {Name} vpc-0fc1ff23f45b678eb
False              {}     vpc-01e23c4a5d6db78e9
False              {Name} vpc-0123456b078b9d01f
False              {}     vpc-12cf3b4f
False              {Name} vpc-0b12d3456a7e8901d
```

- Einzelheiten zur API finden Sie unter DescribeVpcClassicLink AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2VpcClassicLinkDnsSupport`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2VpcClassicLinkDnsSupport`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt den ClassicLink DNS-Unterstützungsstatus von VPCs für die Region eu-west-1

```
Get-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

Ausgabe:

```
ClassicLinkDnsSupported VpcId
----------------------- -----
False                   vpc-0b12d3456a7e8910d
False                   vpc-12cf3b4f
```

- Einzelheiten zur API finden Sie unter DescribeVpcClassicLinkDnsSupportCmdlet-Referenz.AWS Tools for PowerShell

`Get-EC2VpcEndpoint`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2VpcEndpoint`

Beispiel 1: Dieses Beispiel beschreibt einen oder mehrere Ihrer VPC-Endpunkte für die Region eu-west-1. Anschließend leitet es die Ausgabe an den nächsten Befehl weiter, der die VpcEndpointId Eigenschaft auswählt und die Array-VPC-ID als String-Array zurückgibt

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object -ExpandProperty VpcEndpointId
```

Ausgabe:

```
vpce-01a2ab3f4f5cc6f7d
vpce-01d2b345a6787890b
vpce-0012e34d567890e12
vpce-0c123db4567890123
```

Beispiel 2: Dieses Beispiel beschreibt alle VPC-Endpunkte für die Region eu-west-1 und wählt VpcEndpointId,, ServiceName und PrivateDnsEnabled Eigenschaften aus VpcId, um sie in einem tabellarischen Format darzustellen

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object VpcEndpointId, VpcId,
 ServiceName, PrivateDnsEnabled | Format-Table -AutoSize
```

Ausgabe:

```
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
```

Beispiel 3: In diesem Beispiel wird das Richtliniendokument für den VPC-Endpunkt vpce-01a2ab3f4f5cc6f7d in eine JSON-Datei exportiert

```
Get-EC2VpcEndpoint -Region eu-west-1 -VpcEndpointId vpce-01a2ab3f4f5cc6f7d | Select-
Object -expand PolicyDocument | Out-File vpce_policyDocument.json
```

- Einzelheiten AWS Tools for PowerShell zur DescribeVpcEndpointsAPI finden Sie unter Cmdlet-Referenz.

`Get-EC2VpcEndpointService`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2VpcEndpointService`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt den EC2 VPC-Endpunktservice mit dem angegebenen Filter, in diesem Fall com.amazonaws.eu-west-1.ecs. Außerdem wird die Eigenschaft erweitert und die Details werden angezeigt ServiceDetails

```
Get-EC2VpcEndpointService -Region eu-west-1 -MaxResult 5 -Filter @{Name="service-
name";Values="com.amazonaws.eu-west-1.ecs"} | Select-Object -ExpandProperty
 ServiceDetails
```

Ausgabe:

```
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
```

Beispiel 2: In diesem Beispiel werden alle EC2-VPC-Endpunktdienste abgerufen und das ServiceNames passende „ssm“ zurückgegeben

```
Get-EC2VpcEndpointService -Region eu-west-1 | Select-Object -ExpandProperty
 Servicenames | Where-Object { -match "ssm"}
```

Ausgabe:

```
com.amazonaws.eu-west-1.ssm
```

```
com.amazonaws.eu-west-1.ssmmessages
```

- Einzelheiten zur API finden Sie unter DescribeVpcEndpointServicesCmdlet-Referenz.AWS Tools for PowerShell

`Get-EC2VpnConnection`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2VpnConnection`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die angegebene VPN-Verbindung.

```
Get-EC2VpnConnection -VpnConnectionId vpn-12345678
```

Ausgabe:

```
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
```

Beispiel 2: Dieses Beispiel beschreibt jede VPN-Verbindung, deren Status entweder ausstehend oder verfügbar ist.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnConnection -Filter $filter
```

Beispiel 3: Dieses Beispiel beschreibt all Ihre VPN-Verbindungen.

```
Get-EC2VpnConnection
```

- Einzelheiten zur API finden Sie unter DescribeVpnConnections AWS Tools for PowerShellCmdlet-Referenz.

`Get-EC2VpnGateway`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2VpnGateway`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt das angegebene Virtual Private Gateway.

```
Get-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

Ausgabe:

```
AvailabilityZone :
State            : available
Tags             : {}
Type             : ipsec.1
VpcAttachments   : {vpc-12345678}
VpnGatewayId     : vgw-1a2b3c4d
```

Beispiel 2: Dieses Beispiel beschreibt jedes virtuelle private Gateway, dessen Status entweder ausstehend oder verfügbar ist.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnGateway -Filter $filter
```

Beispiel 3: Dieses Beispiel beschreibt alle Ihre virtuellen privaten Gateways.

```
Get-EC2VpnGateway
```

- Einzelheiten zur API finden Sie unter DescribeVpnGateways AWS Tools for PowerShellCmdlet-Referenz.

`Grant-EC2SecurityGroupEgress`

Das folgende Codebeispiel zeigt die Verwendung. `Grant-EC2SecurityGroupEgress`

Tools für PowerShell

Beispiel 1: Dieses Beispiel definiert eine Ausgangsregel für die angegebene Sicherheitsgruppe für EC2-VPC. Die Regel gewährt Zugriff auf den angegebenen IP-Adressbereich am TCP-Port 80. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

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

Beispiel 2: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um das Objekt zu erstellen. IpPermission

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 80
$ip.ToPort = 80
$ip.IpRanges.Add("203.0.113.0/24")

Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Beispiel 3: Dieses Beispiel gewährt Zugriff auf die angegebene Quellsicherheitsgruppe am 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 } )
```

- Einzelheiten zur API finden Sie unter AuthorizeSecurityGroupEgress AWS Tools for PowerShellCmdlet-Referenz.

`Grant-EC2SecurityGroupIngress`

Das folgende Codebeispiel zeigt die Verwendung. `Grant-EC2SecurityGroupIngress`

Beispiel 1: In diesem Beispiel werden Eingangsregeln für eine Sicherheitsgruppe für EC2-VPC definiert. Diese Regeln gewähren Zugriff auf eine bestimmte IP-Adresse für SSH (Port 22) und RDC (Port 3389). Beachten Sie, dass Sie Sicherheitsgruppen für EC2-VPC anhand der Sicherheitsgruppen-ID und nicht anhand des Sicherheitsgruppennamens identifizieren müssen. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$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 )
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um die IpPermission Objekte zu erstellen.

```
$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 )
```

Beispiel 3: Dieses Beispiel definiert Eingangsregeln für eine Sicherheitsgruppe für EC2-Classic. Diese Regeln gewähren Zugriff auf eine bestimmte IP-Adresse für SSH (Port 22) und RDC (Port 3389). Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$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 )
```

Beispiel 4: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um die IpPermission Objekte zu erstellen.

```
$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 )
```

Beispiel 5: Dieses Beispiel gewährt TCP-Port 8081 Zugriff von der angegebenen Quellsicherheitsgruppe (sg-1a2b3c4d) auf die angegebene Sicherheitsgruppe (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 } )
```

Beispiel 6: In diesem Beispiel wird der CIDR 5.5.5.5/32 zu den Eingangsregeln der Sicherheitsgruppe sg-1234abcd für TCP-Port 22-Verkehr mit einer Beschreibung hinzugefügt.

```
$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
```

- Einzelheiten zur AuthorizeSecurityGroupIngressAPI finden Sie unter Cmdlet-Referenz.AWS Tools for PowerShell

`Import-EC2Image`

Das folgende Codebeispiel zeigt die Verwendung. `Import-EC2Image`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Image einer virtuellen Maschine mit einer Festplatte aus dem angegebenen Amazon S3 S3-Bucket mit einem Idempotenz-Token nach Amazon EC2 importiert. Das Beispiel erfordert, dass eine VM-Import-Servicerolle mit dem Standardnamen „vmimport“ vorhanden ist, mit einer Richtlinie, die Amazon EC2 den Zugriff auf den angegebenen Bucket ermöglicht, wie im Thema VM-Importvoraussetzungen erklärt. Um eine benutzerdefinierte Rolle zu verwenden, geben Sie den Rollennamen mithilfe des Parameters an. **`-RoleName`**

```
$container = New-Object Amazon.EC2.Model.ImageDiskContainer
$container.Format="VMDK"
$container.UserBucket = New-Object Amazon.EC2.Model.UserBucket
$container.UserBucket.S3Bucket = "myVirtualMachineImages"
$container.UserBucket.S3Key = "Win_2008_Server_Standard_SP2_64-bit-disk1.vmdk"

$parms = @{
    "ClientToken"="idempotencyToken"
    "Description"="Windows 2008 Standard Image Import"
    "Platform"="Windows"
    "LicenseType"="AWS"
}

Import-EC2Image -DiskContainer $container @parms
```

Ausgabe:

```
Architecture    :
Description     : Windows 2008 Standard Image
Hypervisor      :
ImageId         :
ImportTaskId    : import-ami-abcdefgh
LicenseType     : AWS
Platform        : Windows
Progress        : 2
```

```
SnapshotDetails : {}
Status          : active
StatusMessage   : pending
```

- Einzelheiten zur API finden Sie unter ImportImage AWS Tools for PowerShellCmdlet-Referenz.

`Import-EC2KeyPair`

Das folgende Codebeispiel zeigt die Verwendung. `Import-EC2KeyPair`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein öffentlicher Schlüssel nach EC2 importiert. In der ersten Zeile wird der Inhalt der Datei mit dem öffentlichen Schlüssel (*.pub) in der Variablen gespeichert. **`$publickey`** Als Nächstes konvertiert das Beispiel das UTF8-Format der Datei mit dem öffentlichen Schlüssel in eine Base64-kodierte Zeichenfolge und speichert die konvertierte Zeichenfolge in der Variablen. **`$pkbase64`** In der letzten Zeile wird der konvertierte öffentliche Schlüssel in EC2 importiert. Das Cmdlet gibt den Fingerabdruck und den Namen des Schlüssels als Ergebnisse zurück.

```
$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
```

Ausgabe:

```
KeyFingerprint                                  KeyName
--------------                                  -------
do:d0:15:8f:79:97:12:be:00:fd:df:31:z3:b1:42:z1 Example-user-key
```

- Einzelheiten zur API finden Sie unter ImportKeyPair AWS Tools for PowerShellCmdlet-Referenz.

`Import-EC2Snapshot`

Das folgende Codebeispiel zeigt die Verwendung. `Import-EC2Snapshot`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein VM-Festplatten-Image im Format 'VMDK' in einen Amazon EBS-Snapshot importiert. Das Beispiel erfordert eine VM-Import-Servicerolle mit dem Standardnamen „vmimport“ mit einer Richtlinie, die Amazon EC2 EC2-Zugriff auf den angegebenen Bucket ermöglicht, wie im **`VM Import Prequisites`** Thema unter http://docs.aws.amazon.com/ AWSEC WindowsGuide 2/latest/ /VM .html erklärt. ImportPrerequisites Um eine benutzerdefinierte Rolle zu verwenden, geben Sie den Rollennamen mithilfe des Parameters an. **`-RoleName`**

```
$parms = @{
    "ClientToken"="idempotencyToken"
    "Description"="Disk Image Import"
    "DiskContainer_Description" = "Data disk"
    "DiskContainer_Format" = "VMDK"
    "DiskContainer_S3Bucket" = "myVirtualMachineImages"
    "DiskContainer_S3Key" = "datadiskimage.vmdk"
}

Import-EC2Snapshot @parms
```

Ausgabe:

```
Description            ImportTaskId               SnapshotTaskDetail

-----------------      --------------------       ------------------

Disk Image Import      import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
```

- Einzelheiten zur API finden Sie unter ImportSnapshot AWS Tools for PowerShellCmdlet-Referenz.

`Move-EC2AddressToVpc`

Das folgende Codebeispiel zeigt die Verwendung. `Move-EC2AddressToVpc`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine EC2-Instance mit der öffentlichen IP-Adresse 12.345.67.89 auf die EC2-VPC-Plattform in der Region USA Ost (Nord-Virginia) verschoben.

```
Move-EC2AddressToVpc -PublicIp 12.345.67.89 -Region us-east-1
```

Beispiel 2: In diesem Beispiel werden die Ergebnisse eines Befehls über die Pipeline an das Cmdlet übergeben. Get-EC2Instance Move-EC2AddressToVpc Der Get-EC2Instance Befehl ruft eine Instanz ab, die durch die Instanz-ID angegeben ist, und gibt dann die öffentliche IP-Adresseigenschaft der Instanz zurück.

```
(Get-EC2Instance -Instance i-12345678).Instances.PublicIpAddress | Move-
EC2AddressToVpc
```

- Einzelheiten zur API finden Sie unter MoveAddressToVpc AWS Tools for PowerShellCmdlet-Referenz.

`New-EC2Address`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2Address`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Elastic IP-Adresse zugewiesen, die mit einer Instance in einer VPC verwendet werden soll.

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

Ausgabe:

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

Beispiel 2: In diesem Beispiel wird eine Elastic IP-Adresse zur Verwendung mit einer Instance in EC2-Classic zugewiesen.

```
New-EC2Address
```

Ausgabe:

```
AllocationId         Domain      PublicIp
------------         ------      --------
```

```
                    standard    203.0.113.17
```

- Einzelheiten zur API finden Sie unter AllocateAddressCmdlet-Referenz.AWS Tools for PowerShell

`New-EC2CustomerGateway`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2CustomerGateway`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Kunden-Gateway erstellt.

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

Ausgabe:

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

- Einzelheiten zur API finden Sie unter CreateCustomerGateway AWS Tools for PowerShellCmdlet-Referenz.

`New-EC2DhcpOption`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2DhcpOption`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Satz von DHCP-Optionen erstellt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

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

Ausgabe:

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

Beispiel 2: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um jede DHCP-Option zu erstellen.

```
$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)
```

Ausgabe:

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

- Einzelheiten zur API finden Sie unter CreateDhcpOptionsCmdlet-Referenz.AWS Tools for PowerShell

`New-EC2FlowLog`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2FlowLog`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein EC2-Flowlog für das Subnetz Subnetz-1d234567 zum cloud-watch-log benannten Subnet1-Log für den gesamten REJECT-Traffic mit den Berechtigungen der Rolle „Admin“ erstellt

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

Ausgabe:

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

- Einzelheiten AWS Tools for PowerShell zur API finden Sie unter Cmdlet-Referenz. CreateFlowLogs

`New-EC2Host`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2Host`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird Ihrem Konto ein Dedicated Host für den angegebenen Instance-Typ und die angegebene Verfügbarkeitszone zugewiesen

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

Ausgabe:

```
h-01e23f4cd567890f3
```

- Einzelheiten zur API finden Sie unter AllocateHosts AWS Tools for PowerShellCmdlet-Referenz.

`New-EC2HostReservation`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2HostReservation`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Reservierungsangebot hro-0c1f23456789d0ab mit Konfigurationen erworben, die denen Ihres Dedicated Hosts h-01e23f4cd567890f1 entsprechen

```
New-EC2HostReservation -OfferingId hro-0c1f23456789d0ab HostIdSet
 h-01e23f4cd567890f1
```

Ausgabe:

```
ClientToken       :
CurrencyCode      :
Purchase          : {hr-0123f4b5d67bedc89}
TotalHourlyPrice  : 1.307
TotalUpfrontPrice : 0.000
```

- Einzelheiten zur AWS Tools for PowerShell API finden Sie unter Cmdlet-Referenz. PurchaseHostReservation

`New-EC2Image`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2Image`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird aus der angegebenen Instance ein AMI mit dem angegebenen Namen und der Beschreibung erstellt. Amazon EC2 versucht, die Instance sauber herunterzufahren, bevor das Image erstellt wird, und startet die Instance nach Abschluss neu.

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

Beispiel 2: In diesem Beispiel wird aus der angegebenen Instance ein AMI mit dem angegebenen Namen und der Beschreibung erstellt. Amazon EC2 erstellt das Image, ohne die Instance herunterzufahren und neu zu starten. Daher kann die Dateisystemintegrität des erstellten Images nicht garantiert werden.

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

Beispiel 3: In diesem Beispiel wird ein AMI mit drei Volumes erstellt. Das erste Volume basiert auf einem Amazon EBS-Snapshot. Das zweite Volume ist ein leeres 100-GiB-Amazon-EBS-Volume. Das dritte Volume ist ein Instance-Speicher-Volume. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$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"})
```

- Einzelheiten zur API finden Sie unter CreateImage AWS Tools for PowerShellCmdlet-Referenz.

`New-EC2Instance`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2Instance`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine einzelne Instance des angegebenen AMI in EC2-Classic oder einer Standard-VPC gestartet.

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -InstanceType
 m3.medium -KeyName my-key-pair -SecurityGroup my-security-group
```

Beispiel 2: In diesem Beispiel wird eine einzelne Instance des angegebenen AMI in einer VPC gestartet.

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -SubnetId
 subnet-12345678 -InstanceType t2.micro -KeyName my-key-pair -SecurityGroupId
 sg-12345678
```

Beispiel 3: Um ein EBS-Volume oder ein Instance-Speicher-Volume hinzuzufügen, definieren Sie eine Blockgerätezuordnung und fügen Sie sie dem Befehl hinzu. In diesem Beispiel wird ein Instance-Speicher-Volume hinzugefügt.

```
$bdm = New-Object Amazon.EC2.Model.BlockDeviceMapping
$bdm.VirtualName = "ephemeral0"
$bdm.DeviceName = "/dev/sdf"

New-EC2Instance -ImageId ami-12345678 -BlockDeviceMapping $bdm ...
```

Beispiel 4: Um eines der aktuellen Windows-AMIs anzugeben, rufen Sie dessen AMI-ID mithilfe von abGet-EC2ImageByName. In diesem Beispiel wird eine Instance aus dem aktuellen Basis-AMI für Windows Server 2016 gestartet.

```
$ami = Get-EC2ImageByName WINDOWS_2016_BASE

New-EC2Instance -ImageId $ami.ImageId ...
```

Beispiel 5: Startet eine Instance in der angegebenen dedizierten Host-Umgebung.

```
New-EC2Instance -ImageId ami-1a2b3c4d -InstanceType m4.large -KeyName my-key-pair
 -SecurityGroupId sg-1a2b3c4d  -AvailabilityZone us-west-1a -Tenancy host -HostID
 h-1a2b3c4d5e6f1a2b3
```

Beispiel 6: Diese Anfrage startet zwei Instances und wendet ein Tag mit dem Schlüssel Webserver und dem Wert production auf die Instanzen an. Die Anfrage wendet außerdem ein Tag mit dem Schlüssel cost-center und dem Wert cc123 auf die erstellten Volumes an (in diesem Fall das Root-Volume für jede Instanz).

```
$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
```

- Einzelheiten zur API finden Sie unter RunInstances AWS Tools for PowerShellCmdlet-Referenz.

`New-EC2InstanceExportTask`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2InstanceExportTask`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine **`i-0800b00a00EXAMPLE`** gestoppte Instanz als virtuelle Festplatte (VHD) in den S3-Bucket **`testbucket-export-instances-2019`** exportiert. Die

Zielumgebung ist**`Microsoft`**, und der Regionsparameter wird hinzugefügt, weil sich die Instanz in der **`us-east-1`** Region befindet, während die AWS Standardregion des Benutzers nicht us-east-1 ist. Um den Status der Exportaufgabe abzurufen, kopieren Sie den **`ExportTaskId`** Wert aus den Ergebnissen dieses Befehls und führen Sie dann den Befehl aus **`Get-EC2ExportTask -ExportTaskId export_task_ID_from_results.`**

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

Ausgabe:

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

- Einzelheiten zur API finden Sie unter CreateInstanceExportTask AWS Tools for PowerShellCmdlet-Referenz.

`New-EC2InternetGateway`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2InternetGateway`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Internet-Gateway erstellt.

```
New-EC2InternetGateway
```

Ausgabe:

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

- Einzelheiten zur API finden Sie unter CreateInternetGateway AWS Tools for PowerShellCmdlet-Referenz.

`New-EC2KeyPair`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2KeyPair`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein key pair erstellt und der PEM-kodierte private RSA-Schlüssel in einer Datei mit dem angegebenen Namen erfasst. Wenn Sie verwenden PowerShell, muss die Kodierung auf ASCII eingestellt sein, um einen gültigen Schlüssel zu generieren. Weitere Informationen finden Sie unter Amazon EC2 EC2-Schlüsselpaare erstellen, anzeigen und löschen (https://docs.aws.amazon.com/cli/latest/userguide/ cli-services-ec 2-keypairs.html) im AWS Command Line Interface User Guide.

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

- Einzelheiten zur API finden Sie unter CreateKeyPair AWS Tools for PowerShellCmdlet-Referenz.

`New-EC2NetworkAcl`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2NetworkAcl`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Netzwerk-ACL für die angegebene VPC erstellt.

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

Ausgabe:

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

- Einzelheiten zur API finden Sie unter CreateNetworkAcl AWS Tools for PowerShellCmdlet-Referenz.

`New-EC2NetworkAclEntry`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2NetworkAclEntry`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Eintrag für die angegebene Netzwerk-ACL erstellt. Die Regel erlaubt eingehenden Verkehr von überall (0.0.0.0/0) am UDP-Port 53 (DNS) in jedes zugehörige Subnetz.

```
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
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz. CreateNetworkAclEntryAWS Tools for PowerShell

`New-EC2NetworkInterface`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2NetworkInterface`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerkschnittstelle erstellt.

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

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter CreateNetworkInterface AWS Tools for PowerShellCmdlet-Referenz.

`New-EC2PlacementGroup`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2PlacementGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Platzierungsgruppe mit dem angegebenen Namen erstellt.

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

- Einzelheiten zur API finden Sie unter CreatePlacementGroup AWS Tools for PowerShellCmdlet-Referenz.

`New-EC2Route`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2Route`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Route für die angegebene Routentabelle erstellt. Die Route entspricht dem gesamten Datenverkehr und sendet ihn an das angegebene Internet-Gateway.

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

Ausgabe:

```
True
```

- Einzelheiten zur API finden Sie unter CreateRoute AWS Tools for PowerShellCmdlet-Referenz.

`New-EC2RouteTable`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2RouteTable`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Routentabelle für die angegebene VPC erstellt.

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

Ausgabe:

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

- Einzelheiten zur API finden Sie unter CreateRouteTable AWS Tools for PowerShellCmdlet-Referenz.

`New-EC2ScheduledInstance`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2ScheduledInstance`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene geplante Instanz gestartet.

```
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
```

- Einzelheiten zur API finden Sie unter RunScheduledInstances AWS Tools for PowerShellCmdlet-Referenz.

`New-EC2ScheduledInstancePurchase`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2ScheduledInstancePurchase`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine geplante Instance gekauft.

```
$request = New-Object Amazon.EC2.Model.PurchaseRequest
$request.InstanceCount = 1
$request.PurchaseToken = "eyJ2IjoiMSIsInMiOjEsImMiOi..."
New-EC2ScheduledInstancePurchase -PurchaseRequest $request
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter PurchaseScheduledInstances AWS Tools for PowerShellCmdlet-Referenz.

`New-EC2SecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2SecurityGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Sicherheitsgruppe für die angegebene VPC erstellt.

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

Ausgabe:

```
sg-12345678
```

Beispiel 2: In diesem Beispiel wird eine Sicherheitsgruppe für EC2-Classic erstellt.

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

Ausgabe:

```
sg-45678901
```

- Einzelheiten zur API finden Sie unter CreateSecurityGroup AWS Tools for PowerShellCmdlet-Referenz.

`New-EC2Snapshot`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2Snapshot`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Snapshot des angegebenen Volumes erstellt.

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

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter CreateSnapshot AWS Tools for PowerShellCmdlet-Referenz.

`New-EC2SpotDatafeedSubscription`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2SpotDatafeedSubscription`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Spot-Instance-Datenfeed erstellt.

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

Ausgabe:

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

- Einzelheiten zur API finden Sie unter CreateSpotDatafeedSubscription AWS Tools for PowerShellCmdlet-Referenz.

`New-EC2Subnet`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2Subnet`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Subnetz mit dem angegebenen CIDR erstellt.

```
New-EC2Subnet -VpcId vpc-12345678 -CidrBlock 10.0.0.0/24
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter CreateSubnet AWS Tools for PowerShellCmdlet-Referenz.

`New-EC2Tag`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2Tag`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der angegebenen Ressource ein einzelnes Tag hinzugefügt. Der Tag-Schlüssel ist 'myTag' und der Tag-Wert ist 'myTagValue'. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
New-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag"; Value="myTagValue" }
```

Beispiel 2: In diesem Beispiel werden die angegebenen Tags der angegebenen Ressource aktualisiert oder hinzugefügt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
New-EC2Tag -Resource i-12345678 -Tag @( @{ Key="myTag"; Value="newTagValue" },
 @{ Key="test"; Value="anotherTagValue" } )
```

Beispiel 3: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um das Tag für den Tag-Parameter zu erstellen.

```
$tag = New-Object Amazon.EC2.Model.Tag
```

```
$tag.Key = "myTag"
$tag.Value = "myTagValue"

New-EC2Tag -Resource i-12345678 -Tag $tag
```

- Einzelheiten zur API finden Sie unter CreateTags AWS Tools for PowerShellCmdlet-Referenz.

`New-EC2Volume`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2Volume`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Volumen erstellt.

```
New-EC2Volume -Size 50 -AvailabilityZone us-west-2a -VolumeType gp2
```

Ausgabe:

```
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
```

Beispiel 2: Diese Beispielanforderung erstellt ein Volume und wendet ein Tag mit einem Stack-Schlüssel und einem Produktionswert an.

```
$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
```

- Einzelheiten zur API finden Sie unter CreateVolume AWS Tools for PowerShellCmdlet-Referenz.

`New-EC2Vpc`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2Vpc`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine VPC mit dem angegebenen CIDR erstellt. Amazon VPC erstellt außerdem Folgendes für die VPC: einen Standard-DHCP-Optionssatz, eine Haupt-Routing-Tabelle und eine Standard-Netzwerk-ACL.

```
New-EC2VPC -CidrBlock 10.0.0.0/16
```

Ausgabe:

```
CidrBlock       : 10.0.0.0/16
DhcpOptionsId   : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault       : False
State           : pending
Tags            : {}
VpcId           : vpc-12345678
```

- Einzelheiten zur API finden Sie unter CreateVpcCmdlet-Referenz.AWS Tools for PowerShell

`New-EC2VpcEndpoint`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2VpcEndpoint`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein neuer VPC-Endpunkt für den Service com.amazonaws.eu-west-1.s3 in der VPC vpc-0fc1ff23f45b678eb erstellt

```
New-EC2VpcEndpoint -ServiceName com.amazonaws.eu-west-1.s3 -VpcId
 vpc-0fc1ff23f45b678eb
```

Ausgabe:

```
ClientToken VpcEndpoint
----------- -----------
            Amazon.EC2.Model.VpcEndpoint
```

- Einzelheiten zur AWS Tools for PowerShell API finden CreateVpcEndpointSie unter Cmdlet-Referenz.

`New-EC2VpnConnection`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2VpnConnection`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine VPN-Verbindung zwischen dem angegebenen Virtual Private Gateway und dem angegebenen Kunden-Gateway erstellt. Die Ausgabe enthält die Konfigurationsinformationen, die Ihr Netzwerkadministrator benötigt, im XML-Format.

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d
```

Ausgabe:

```
CustomerGatewayConfiguration : [XML document]
CustomerGatewayId            : cgw-1a2b3c4d
Options                      :
Routes                       : {}
State                        : pending
Tags                         : {}
Type                         :
VgwTelemetry                 : {}
VpnConnectionId              : vpn-12345678
VpnGatewayId                 : vgw-1a2b3c4d
```

Beispiel 2: In diesem Beispiel wird die VPN-Verbindung hergestellt und die Konfiguration in einer Datei mit dem angegebenen Namen erfasst.

```
(New-EC2VpnConnection -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d).CustomerGatewayConfiguration | Out-File C:\path\vpn-configuration.xml
```

Beispiel 3: In diesem Beispiel wird eine VPN-Verbindung mit statischem Routing zwischen dem angegebenen virtuellen privaten Gateway und dem angegebenen Kunden-Gateway erstellt.

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d -Options_StaticRoutesOnly $true
```

- Einzelheiten zur API finden Sie unter CreateVpnConnection AWS Tools for PowerShellCmdlet-Referenz.

`New-EC2VpnConnectionRoute`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2VpnConnectionRoute`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene statische Route für die angegebene VPN-Verbindung erstellt.

```
New-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock
 11.12.0.0/16
```

- Einzelheiten zur API finden Sie unter CreateVpnConnectionRoute AWS Tools for PowerShellCmdlet-Referenz.

`New-EC2VpnGateway`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2VpnGateway`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene virtuelle private Gateway erstellt.

```
New-EC2VpnGateway -Type ipsec.1
```

Ausgabe:

```
AvailabilityZone :
State            : available
Tags             : {}
Type             : ipsec.1
```

```
VpcAttachments   : {}
VpnGatewayId     : vgw-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter CreateVpnGateway AWS Tools for PowerShellCmdlet-Referenz.

`Register-EC2Address`

Das folgende Codebeispiel zeigt die Verwendung. `Register-EC2Address`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Elastic IP-Adresse der angegebenen Instance in einer VPC zugeordnet.

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

Ausgabe:

```
eipassoc-12345678
```

Beispiel 2: In diesem Beispiel wird die angegebene Elastic IP-Adresse der angegebenen Instance in EC2-Classic zugeordnet.

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

- Einzelheiten zur API finden Sie unter AssociateAddress AWS Tools for PowerShellCmdlet-Referenz.

`Register-EC2DhcpOption`

Das folgende Codebeispiel zeigt die Verwendung. `Register-EC2DhcpOption`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene DHCP-Optionssatz der angegebenen VPC zugeordnet.

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

Beispiel 2: In diesem Beispiel werden die standardmäßigen DHCP-Optionen der angegebenen VPC zugeordnet.

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

- Einzelheiten zur API finden Sie unter AssociateDhcpOptions AWS Tools for PowerShellCmdlet-Referenz.

`Register-EC2Image`

Das folgende Codebeispiel zeigt die Verwendung. `Register-EC2Image`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein AMI mithilfe der angegebenen Manifestdatei in Amazon S3 registriert.

```
Register-EC2Image -ImageLocation my-s3-bucket/my-web-server-ami/image.manifest.xml -
Name my-web-server-ami
```

- Einzelheiten zur API finden Sie unter RegisterImage AWS Tools for PowerShellCmdlet-Referenz.

`Register-EC2PrivateIpAddress`

Das folgende Codebeispiel zeigt die Verwendung. `Register-EC2PrivateIpAddress`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene sekundäre private IP-Adresse der angegebenen Netzwerkschnittstelle zugewiesen.

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

Beispiel 2: In diesem Beispiel werden zwei sekundäre private IP-Adressen erstellt und der angegebenen Netzwerkschnittstelle zugewiesen.

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

- Einzelheiten zur API finden Sie unter AssignPrivateIpAddresses AWS Tools for PowerShellCmdlet-Referenz.

`Register-EC2RouteTable`

Das folgende Codebeispiel zeigt die Verwendung. `Register-EC2RouteTable`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Routing-Tabelle dem angegebenen Subnetz zugeordnet.

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

Ausgabe:

```
rtbassoc-12345678
```

- Einzelheiten zur API finden Sie unter AssociateRouteTable AWS Tools for PowerShellCmdlet-Referenz.

`Remove-EC2Address`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2Address`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Elastic IP-Adresse für Instances in einer VPC veröffentlicht.

```
Remove-EC2Address -AllocationId eipalloc-12345678 -Force
```

Beispiel 2: In diesem Beispiel wird die angegebene Elastic IP-Adresse für Instances in EC2-Classic veröffentlicht.

```
Remove-EC2Address -PublicIp 198.51.100.2 -Force
```

- Einzelheiten zur API finden Sie unter ReleaseAddress AWS Tools for PowerShellCmdlet-Referenz.

`Remove-EC2CapacityReservation`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2CapacityReservation`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Kapazitätsreservierung cr-0c1f2345db6f7cdba storniert

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

Ausgabe:

```
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
```

- Einzelheiten zur API CancelCapacityReservationfinden AWS Tools for PowerShell Sie unter Cmdlet-Referenz.

`Remove-EC2CustomerGateway`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2CustomerGateway`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Kunden-Gateway gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

Ausgabe:

```
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"):
```

- Einzelheiten zur API finden Sie unter DeleteCustomerGateway AWS Tools for PowerShellCmdlet-Referenz.

`Remove-EC2DhcpOption`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2DhcpOption`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene DHCP-Optionssatz gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d
```

Ausgabe:

```
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"):
```

- Einzelheiten zur API finden Sie unter DeleteDhcpOptions AWS Tools for PowerShellCmdlet-Referenz.

`Remove-EC2FlowLog`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2FlowLog`

Tools für PowerShell

Beispiel 1: Dieses Beispiel entfernt den angegebenen Wert FlowLogId fl-01a2b3456a789c01

```
Remove-EC2FlowLog -FlowLogId fl-01a2b3456a789c01
```

Ausgabe:

```
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
```

- Einzelheiten zur API DeleteFlowLogsfinden AWS Tools for PowerShell Sie unter Cmdlet-Referenz.

`Remove-EC2Host`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2Host`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Host-ID h-0badafd1dcb2f3456 veröffentlicht

```
Remove-EC2Host -HostId h-0badafd1dcb2f3456
```

Ausgabe:

```
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} {}
```

- Einzelheiten AWS Tools for PowerShell zur ReleaseHostsAPI finden Sie unter Cmdlet-Referenz.

`Remove-EC2Instance`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2Instance`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instanz beendet (die Instanz läuft möglicherweise oder befindet sich im Status „gestoppt“). Das Cmdlet fordert Sie zur Bestätigung auf, bevor Sie fortfahren. Verwenden Sie die Befehlszeilenoption -Force, um die Aufforderung zu unterdrücken.

```
Remove-EC2Instance -InstanceId i-12345678
```

Ausgabe:

```
CurrentState                    InstanceId    PreviousState
------------                    ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

- Einzelheiten zur API finden Sie unter TerminateInstancesCmdlet-Referenz.AWS Tools for PowerShell

`Remove-EC2InternetGateway`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2InternetGateway`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Internet-Gateway gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

Ausgabe:

```
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"):
```

- Einzelheiten zur API finden Sie unter DeleteInternetGateway AWS Tools for PowerShellCmdlet-Referenz.

`Remove-EC2KeyPair`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2KeyPair`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene key pair gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2KeyPair -KeyName my-key-pair
```

Ausgabe:

```
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"):
```

- Einzelheiten zur API finden Sie unter DeleteKeyPair AWS Tools for PowerShellCmdlet-Referenz.

`Remove-EC2NetworkAcl`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2NetworkAcl`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerk-ACL gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2NetworkAcl -NetworkAclId acl-12345678
```

Ausgabe:

```
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"):
```

- Einzelheiten zur API finden Sie unter DeleteNetworkAcl AWS Tools for PowerShellCmdlet-Referenz.

`Remove-EC2NetworkAclEntry`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2NetworkAclEntry`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Regel aus der angegebenen Netzwerk-ACL entfernt. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100
```

Ausgabe:

```
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"):
```

- Einzelheiten zur API finden Sie unter DeleteNetworkAclEntry AWS Tools for PowerShellCmdlet-Referenz.

`Remove-EC2NetworkInterface`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2NetworkInterface`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerkschnittstelle gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

Ausgabe:

```
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"):
```

- Einzelheiten zur API finden Sie unter DeleteNetworkInterface AWS Tools for PowerShellCmdlet-Referenz.

`Remove-EC2PlacementGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2PlacementGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Platzierungsgruppe gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2PlacementGroup -GroupName my-placement-group
```

Ausgabe:

```
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"):
```

- Einzelheiten zur API finden Sie unter DeletePlacementGroup AWS Tools for PowerShellCmdlet-Referenz.

`Remove-EC2Route`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2Route`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Route aus der angegebenen Routentabelle gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0
```

Ausgabe:

```
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"):
```

- Einzelheiten zur API finden Sie unter DeleteRoute AWS Tools for PowerShellCmdlet-Referenz.

`Remove-EC2RouteTable`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2RouteTable`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Routentabelle gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

Ausgabe:

```
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"):
```

- Einzelheiten zur API finden Sie unter DeleteRouteTable AWS Tools for PowerShellCmdlet-Referenz.

`Remove-EC2SecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2SecurityGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Sicherheitsgruppe für EC2-VPC gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2SecurityGroup -GroupId sg-12345678
```

Ausgabe:

```
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"):
```

Beispiel 2: In diesem Beispiel wird die angegebene Sicherheitsgruppe für EC2-Classic gelöscht.

```
Remove-EC2SecurityGroup -GroupName my-security-group -Force
```

- Einzelheiten zur API finden Sie unter DeleteSecurityGroupCmdlet-Referenz.AWS Tools for PowerShell

`Remove-EC2Snapshot`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2Snapshot`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Snapshot gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2Snapshot -SnapshotId snap-12345678
```

Ausgabe:

```
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"):
```

- Einzelheiten zur API finden Sie unter DeleteSnapshot AWS Tools for PowerShellCmdlet-Referenz.

`Remove-EC2SpotDatafeedSubscription`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2SpotDatafeedSubscription`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird Ihr Spot-Instance-Datenfeed gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2SpotDatafeedSubscription
```

Ausgabe:

```
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"):
```

- Einzelheiten zur API finden Sie unter DeleteSpotDatafeedSubscription AWS Tools for PowerShellCmdlet-Referenz.

`Remove-EC2Subnet`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2Subnet`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Subnetz gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2Subnet -SubnetId subnet-1a2b3c4d
```

Ausgabe:

```
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"):
```

- Einzelheiten zur API finden Sie unter DeleteSubnet AWS Tools for PowerShellCmdlet-Referenz.

`Remove-EC2Tag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2Tag`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Tag unabhängig vom Tag-Wert aus der angegebenen Ressource gelöscht. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag" } -Force
```

Beispiel 2: In diesem Beispiel wird das angegebene Tag aus der angegebenen Ressource gelöscht, aber nur, wenn der Tag-Wert übereinstimmt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag";Value="myTagValue" } -Force
```

Beispiel 3: In diesem Beispiel wird das angegebene Tag unabhängig vom Tag-Wert aus der angegebenen Ressource gelöscht.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

Beispiel 4: In diesem Beispiel wird das angegebene Tag aus der angegebenen Ressource gelöscht, aber nur, wenn der Tag-Wert übereinstimmt.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

- Einzelheiten zur API finden Sie unter DeleteTags AWS Tools for PowerShellCmdlet-Referenz.

`Remove-EC2Volume`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2Volume`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Volume getrennt. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2Volume -VolumeId vol-12345678
```

Ausgabe:

```
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"):
```

- Einzelheiten zur API finden Sie unter DeleteVolume AWS Tools for PowerShellCmdlet-Referenz.

`Remove-EC2Vpc`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2Vpc`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene VPC gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2Vpc -VpcId vpc-12345678
```

Ausgabe:

```
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"):
```

- Einzelheiten zur API finden Sie unter DeleteVpc AWS Tools for PowerShellCmdlet-Referenz.

`Remove-EC2VpnConnection`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2VpnConnection`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene VPN-Verbindung gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2VpnConnection -VpnConnectionId vpn-12345678
```

Ausgabe:

```
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"):
```

- Einzelheiten zur API finden Sie unter DeleteVpnConnection AWS Tools for PowerShellCmdlet-Referenz.

`Remove-EC2VpnConnectionRoute`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2VpnConnectionRoute`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene statische Route aus der angegebenen VPN-Verbindung entfernt. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock
 11.12.0.0/16
```

Ausgabe:

```
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"):
```

- Einzelheiten zur API finden Sie unter DeleteVpnConnectionRoute AWS Tools for PowerShellCmdlet-Referenz.

`Remove-EC2VpnGateway`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2VpnGateway`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene virtuelle private Gateway gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

Ausgabe:

```
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"):
```

- Einzelheiten zur API finden Sie unter DeleteVpnGateway AWS Tools for PowerShellCmdlet-Referenz.

`Request-EC2SpotFleet`

Das folgende Codebeispiel zeigt die Verwendung. `Request-EC2SpotFleet`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Spot-Flottenanfrage in der Availability Zone mit dem niedrigsten Preis für den angegebenen Instance-Typ erstellt. Wenn Ihr Konto nur EC2-VPC unterstützt, startet die Spot-Flotte die Instances in der Availability Zone mit dem niedrigsten Preis, die über ein Standardsubnetz verfügt. Wenn Ihr Konto EC2-Classic unterstützt, startet die Spot-Flotte die Instances in EC2-Classic in der Availability Zone mit dem niedrigsten Preis. Beachten Sie, dass der Preis, den Sie zahlen, den angegebenen Spot-Preis für die Anfrage nicht überschreiten wird.

```
$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
```

- Einzelheiten zur API finden Sie unter RequestSpotFleet AWS Tools for PowerShellCmdlet-Referenz.

`Request-EC2SpotInstance`

Das folgende Codebeispiel zeigt die Verwendung. `Request-EC2SpotInstance`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine einmalige Spot-Instance im angegebenen Subnetz angefordert. Beachten Sie, dass die Sicherheitsgruppe für die VPC erstellt werden muss, die das angegebene Subnetz enthält, und dass sie über die Netzwerkschnittstelle anhand der ID angegeben werden muss. Wenn Sie eine Netzwerkschnittstelle angeben, müssen Sie die Subnetz-ID mithilfe der Netzwerkschnittstelle angeben.

```
$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
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter RequestSpotInstances AWS Tools for PowerShellCmdlet-Referenz.

`Reset-EC2ImageAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Reset-EC2ImageAttribute`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Attribut 'launchPermission' auf seinen Standardwert zurückgesetzt. Standardmäßig sind AMIs privat.

```
Reset-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

- Einzelheiten zur API finden Sie unter ResetImageAttribute AWS Tools for PowerShellCmdlet-Referenz.

`Reset-EC2InstanceAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Reset-EC2InstanceAttribute`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Attribut 'sriovNetSupport' für die angegebene Instanz zurückgesetzt.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

Beispiel 2: In diesem Beispiel wird das Attribut 'ebsOptimized' für die angegebene Instance zurückgesetzt.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

Beispiel 3: In diesem Beispiel wird das Attribut 'sourceDestCheck' für die angegebene Instance zurückgesetzt.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sourceDestCheck
```

Beispiel 4: In diesem Beispiel wird das Attribut 'disableApiTermination' für die angegebene Instanz zurückgesetzt.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

Beispiel 5: In diesem Beispiel wird das Attribut 'instanceInitiatedShutdownBehavior' für die angegebene Instanz zurückgesetzt.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
 instanceInitiatedShutdownBehavior
```

- Einzelheiten zur API finden Sie unter ResetInstanceAttributeCmdlet-Referenz.AWS Tools for PowerShell

`Reset-EC2NetworkInterfaceAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Reset-EC2NetworkInterfaceAttribute`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Quell-/Zielüberprüfung für die angegebene Netzwerkschnittstelle zurückgesetzt.

```
Reset-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -SourceDestCheck
```

- Einzelheiten zur API finden Sie unter ResetNetworkInterfaceAttributeCmdlet-Referenz.AWS Tools for PowerShell

`Reset-EC2SnapshotAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Reset-EC2SnapshotAttribute`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Attribut des angegebenen Snapshots zurückgesetzt.

```
Reset-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission
```

- Einzelheiten zur API finden Sie unter ResetSnapshotAttribute AWS Tools for PowerShellCmdlet-Referenz.

`Restart-EC2Instance`

Das folgende Codebeispiel zeigt die Verwendung. `Restart-EC2Instance`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instanz neu gestartet.

```
Restart-EC2Instance -InstanceId i-12345678
```

- Einzelheiten zur API finden Sie unter RebootInstances AWS Tools for PowerShellCmdlet-Referenz.

`Revoke-EC2SecurityGroupEgress`

Das folgende Codebeispiel zeigt die Verwendung. `Revoke-EC2SecurityGroupEgress`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Regel für die angegebene Sicherheitsgruppe für EC2-VPC entfernt. Dadurch wird der Zugriff auf den angegebenen IP-Adressbereich am TCP-Port 80 aufgehoben. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

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

Beispiel 2: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um das Objekt zu erstellen. IpPermission

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 80
$ip.ToPort = 80
$ip.IpRanges.Add("203.0.113.0/24")
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Beispiel 3: In diesem Beispiel wird der Zugriff auf die angegebene Quellsicherheitsgruppe am TCP-Port 80 gesperrt.

```
$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 } )
```

- Einzelheiten zur API finden Sie unter RevokeSecurityGroupEgress AWS Tools for PowerShellCmdlet-Referenz.

`Revoke-EC2SecurityGroupIngress`

Das folgende Codebeispiel zeigt die Verwendung. `Revoke-EC2SecurityGroupIngress`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der Zugriff auf TCP-Port 22 aus dem angegebenen Adressbereich für die angegebene Sicherheitsgruppe für EC2-VPC VPC. Beachten Sie, dass Sie Sicherheitsgruppen für EC2-VPC anhand der Sicherheitsgruppen-ID und nicht anhand des Sicherheitsgruppennamens identifizieren müssen. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.0/24" }
Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um das Objekt zu erstellen. IpPermission

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 22
```

```
$ip.ToPort = 22
$ip.IpRanges.Add("203.0.113.0/24")

Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

Beispiel 3: In diesem Beispiel wird der Zugriff auf TCP-Port 22 aus dem angegebenen Adressbereich für die angegebene Sicherheitsgruppe für EC2-Classic gesperrt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.0/24" }

Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

Beispiel 4: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um das Objekt zu erstellen. IpPermission

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 22
$ip.ToPort = 22
$ip.IpRanges.Add("203.0.113.0/24")

Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

- Einzelheiten zur API finden Sie unter RevokeSecurityGroupIngress AWS Tools for PowerShellCmdlet-Referenz.

`Send-EC2InstanceStatus`

Das folgende Codebeispiel zeigt die Verwendung. `Send-EC2InstanceStatus`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird Statusfeedback für die angegebene Instanz gemeldet.

```
Send-EC2InstanceStatus -Instance i-12345678 -Status impaired -ReasonCode
 unresponsive
```

- Einzelheiten zur API finden Sie unter ReportInstanceStatus AWS Tools for PowerShellCmdlet-Referenz.

`Set-EC2NetworkAclAssociation`

Das folgende Codebeispiel zeigt die Verwendung. `Set-EC2NetworkAclAssociation`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerk-ACL dem Subnetz für die angegebene Netzwerk-ACL-Zuordnung zugeordnet.

```
Set-EC2NetworkAclAssociation -NetworkAclId acl-12345678 -AssociationId
 aclassoc-1a2b3c4d
```

Ausgabe:

```
aclassoc-87654321
```

- Einzelheiten zur API finden Sie unter ReplaceNetworkAclAssociation AWS Tools for PowerShellCmdlet-Referenz.

`Set-EC2NetworkAclEntry`

Das folgende Codebeispiel zeigt die Verwendung. `Set-EC2NetworkAclEntry`

Tools für PowerShell

Beispiel 1: Dieses Beispiel ersetzt den angegebenen Eintrag für die angegebene Netzwerk-ACL. Die neue Regel erlaubt eingehenden Verkehr von der angegebenen Adresse zu jedem zugehörigen Subnetz.

```
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
```

- Einzelheiten zur API finden Sie unter ReplaceNetworkAclEntry AWS Tools for PowerShellCmdlet-Referenz.

`Set-EC2Route`

Das folgende Codebeispiel zeigt die Verwendung. `Set-EC2Route`

Tools für PowerShell

Beispiel 1: Dieses Beispiel ersetzt die angegebene Route für die angegebene Routentabelle. Die neue Route sendet den angegebenen Verkehr an das angegebene Virtual Private Gateway.

```
Set-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 10.0.0.0/24 -GatewayId
 vgw-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter ReplaceRoute AWS Tools for PowerShellCmdlet-Referenz.

`Set-EC2RouteTableAssociation`

Das folgende Codebeispiel zeigt die Verwendung. `Set-EC2RouteTableAssociation`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Routing-Tabelle dem Subnetz für die angegebene Routentabellenzuordnung zugeordnet.

```
Set-EC2RouteTableAssociation -RouteTableId rtb-1a2b3c4d -AssociationId
 rtbassoc-12345678
```

Ausgabe:

```
rtbassoc-87654321
```

- Einzelheiten zur API finden Sie unter ReplaceRouteTableAssociation AWS Tools for PowerShellCmdlet-Referenz.

`Start-EC2Instance`

Das folgende Codebeispiel zeigt die Verwendung. `Start-EC2Instance`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instanz gestartet.

```
Start-EC2Instance -InstanceId i-12345678
```

Ausgabe:

```
CurrentState                      InstanceId    PreviousState
------------                      ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

Beispiel 2: In diesem Beispiel werden die angegebenen Instanzen gestartet.

```
@("i-12345678", "i-76543210") | Start-EC2Instance
```

Beispiel 3: In diesem Beispiel werden die Instanzen gestartet, die derzeit gestoppt sind. Die von zurückgegebenen Instanzobjekte Get-EC2Instance werden über die Pipeline an Start-EC2Instance übergeben. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
(Get-EC2Instance -Filter @{ Name="instance-state-name"; Values="stopped"}).Instances
 | Start-EC2Instance
```

Beispiel 4: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um den Filter für den Filter-Parameter zu erstellen.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "instance-state-name"
$filter.Values = "stopped"

(Get-EC2Instance -Filter $filter).Instances | Start-EC2Instance
```

- Einzelheiten zur API finden Sie unter StartInstances AWS Tools for PowerShellCmdlet-Referenz.

`Start-EC2InstanceMonitoring`

Das folgende Codebeispiel zeigt die Verwendung. `Start-EC2InstanceMonitoring`

Tools für PowerShell

Beispiel 1: Dieses Beispiel ermöglicht eine detaillierte Überwachung für die angegebene Instanz.

```
Start-EC2InstanceMonitoring -InstanceId i-12345678
```

Ausgabe:

```
InstanceId    Monitoring
----------    ----------
i-12345678    Amazon.EC2.Model.Monitoring
```

- Einzelheiten zur API finden Sie unter MonitorInstances AWS Tools for PowerShellCmdlet-Referenz.

`Stop-EC2ImportTask`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-EC2ImportTask`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Importaufgabe (entweder Snapshot- oder Bildimport) abgebrochen. Falls erforderlich, kann mithilfe des **`-CancelReason`** Parameters ein Grund angegeben werden.

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

- Einzelheiten zur API finden Sie unter CancelImportTask AWS Tools for PowerShellCmdlet-Referenz.

`Stop-EC2Instance`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-EC2Instance`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instanz gestoppt.

```
Stop-EC2Instance -InstanceId i-12345678
```

Ausgabe:

```
CurrentState                    InstanceId    PreviousState
------------                    ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

- Einzelheiten zur API finden Sie unter StopInstances AWS Tools for PowerShellCmdlet-Referenz.

`Stop-EC2InstanceMonitoring`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-EC2InstanceMonitoring`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die detaillierte Überwachung für die angegebene Instanz deaktiviert.

```
Stop-EC2InstanceMonitoring -InstanceId i-12345678
```

Ausgabe:

```
InstanceId    Monitoring
----------    ----------
i-12345678    Amazon.EC2.Model.Monitoring
```

- Einzelheiten zur API finden Sie unter UnmonitorInstances AWS Tools for PowerShellCmdlet-Referenz.

`Stop-EC2SpotFleetRequest`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-EC2SpotFleetRequest`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Spot-Flottenanforderung storniert und die zugehörigen Spot-Instances beendet.

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

Beispiel 2: In diesem Beispiel wird die angegebene Spot-Flottenanforderung storniert, ohne die zugehörigen Spot-Instances zu beenden.

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

- Einzelheiten zur API finden Sie unter CancelSpotFleetRequests AWS Tools for PowerShellCmdlet-Referenz.

`Stop-EC2SpotInstanceRequest`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-EC2SpotInstanceRequest`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Spot-Instance-Anfrage storniert.

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

Ausgabe:

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

- Einzelheiten zur API finden Sie unter CancelSpotInstanceRequests AWS Tools for PowerShellCmdlet-Referenz.

`Unregister-EC2Address`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-EC2Address`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Elastic IP-Adresse von der angegebenen Instance in einer VPC getrennt.

```
Unregister-EC2Address -AssociationId eipassoc-12345678
```

Beispiel 2: In diesem Beispiel wird die Zuordnung der angegebenen Elastic IP-Adresse zur angegebenen Instance in EC2-Classic aufgehoben.

```
Unregister-EC2Address -PublicIp 203.0.113.17
```

- Einzelheiten zur API finden Sie unter DisassociateAddressCmdlet-Referenz.AWS Tools for PowerShell

`Unregister-EC2Image`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-EC2Image`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Registrierung des angegebenen AMI aufgehoben.

```
Unregister-EC2Image -ImageId ami-12345678
```

- Einzelheiten zur API finden Sie unter DeregisterImage AWS Tools for PowerShellCmdlet-Referenz.

`Unregister-EC2PrivateIpAddress`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-EC2PrivateIpAddress`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Zuweisung der angegebenen privaten IP-Adresse zur angegebenen Netzwerkschnittstelle aufgehoben.

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

- Einzelheiten zur API finden Sie unter UnassignPrivateIpAddresses AWS Tools for PowerShellCmdlet-Referenz.

`Unregister-EC2RouteTable`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-EC2RouteTable`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Zuordnung zwischen einer Routing-Tabelle und einem Subnetz entfernt.

```
Unregister-EC2RouteTable -AssociationId rtbassoc-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter DisassociateRouteTable AWS Tools for PowerShellCmdlet-Referenz.

Amazon ECR-Beispiele mit Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon ECR Aktionen ausführen und allgemeine Szenarien implementieren. AWS Tools for PowerShell

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

Aktionen

`Get-ECRLoginCommand`

Das folgende Codebeispiel zeigt die Verwendung`Get-ECRLoginCommand`.

Tools für PowerShell

Beispiel 1: Gibt ein PSObject zurück, das Anmeldeinformationen enthält, die zur Authentifizierung bei jeder Amazon ECR-Registrierung verwendet werden können, auf die Ihr IAM-Prinzipal Zugriff hat. Die Anmeldeinformationen und der Regionsendpunkt, die für den Aufruf zum Abrufen des Autorisierungstokens erforderlich sind, stammen aus den Shell-Standardeinstellungen (die mit den Cmdlets or eingerichtet wurden). **`Set-AWSCredential/Set-DefaultAWSRegion Initialize-AWSDefaultConfiguration`** Sie können die Command-Eigenschaft mit Invoke-Expression verwenden, um sich bei der angegebenen Registrierung anzumelden, oder die zurückgegebenen Anmeldeinformationen in anderen Tools verwenden, für die eine Anmeldung erforderlich ist.

```
Get-ECRLoginCommand
```

Ausgabe:

```
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
```

Beispiel 2: Ruft ein PSObject ab, das Anmeldeinformationen enthält, die Sie als Eingabe für einen Docker-Login-Befehl verwenden. Sie können eine beliebige Amazon ECR-Registry-URI für die Authentifizierung angeben, solange Ihr IAM-Prinzipal Zugriff auf diese Registrierung hat.

```
(Get-ECRLoginCommand).Password | docker login --username AWS --password-stdin
 012345678910.dkr.ecr.us-east-1.amazonaws.com
```

- API-Details finden Sie unter Get-ECR in der Cmdlet-Referenz LoginCommand.AWS Tools for PowerShell

Amazon ECS-Beispiele für die Verwendung von Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon ECS Aktionen ausführen und allgemeine Szenarien implementieren. AWS Tools for PowerShell

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`Get-ECSClusterDetail`

Das folgende Codebeispiel zeigt die Verwendung`Get-ECSClusterDetail`.

Tools für PowerShell

Beispiel 1: Dieses Cmdlet beschreibt einen oder mehrere Ihrer ECS-Cluster.

```
Get-ECSClusterDetail -Cluster "LAB-ECS-CL" -Include SETTINGS | Select-Object *
```

Ausgabe:

```
LoggedAt         : 12/27/2019 9:27:41 PM
Clusters         : {LAB-ECS-CL}
Failures         : {}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 396
HttpStatusCode   : OK
```

- Einzelheiten zur API finden Sie unter DescribeClusters AWS Tools for PowerShellCmdlet-Referenz.

`Get-ECSClusterList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ECSClusterList`

Tools für PowerShell

Beispiel 1: Dieses Cmdlet gibt eine Liste vorhandener ECS-Cluster zurück.

```
Get-ECSClusterList
```

Ausgabe:

```
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS-CL
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS
```

- Einzelheiten zur API finden Sie unter ListClusters AWS Tools for PowerShellCmdlet-Referenz.

`Get-ECSClusterService`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ECSClusterService`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden alle Dienste aufgeführt, die in Ihrem Standardcluster ausgeführt werden.

```
Get-ECSClusterService
```

Beispiel 2: In diesem Beispiel werden alle Dienste aufgeführt, die im angegebenen Cluster ausgeführt werden.

```
Get-ECSClusterService -Cluster myCluster
```

Beispiel 3: In diesem Beispiel werden die Dienste aufgeführt, die im angegebenen Cluster ausgeführt werden, wobei maximal 10 Dienstdetails gleichzeitig abgerufen werden.

```
$nextToken = $null
do
{
    Get-ECSClusterService -Cluster myCluster -MaxResult 10 -NextToken $nextToken
    $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Einzelheiten zur API finden Sie unter ListServices AWS Tools for PowerShellCmdlet-Referenz.

`Get-ECSService`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ECSService`

Tools für PowerShell

Beispiel 1: Dieses Beispiel zeigt, wie Sie Details zu einem bestimmten Dienst aus Ihrem Standardcluster abrufen.

```
Get-ECSService -Service my-hhtp-service
```

Beispiel 2: Dieses Beispiel zeigt, wie Sie Details zu einem bestimmten Dienst abrufen, der im benannten Cluster ausgeführt wird.

```
Get-ECSService -Cluster myCluster -Service my-hhtp-service
```

- Einzelheiten zur API finden Sie unter DescribeServices AWS Tools for PowerShellCmdlet-Referenz.

`New-ECSCluster`

Das folgende Codebeispiel zeigt die Verwendung. `New-ECSCluster`

Tools für PowerShell

Beispiel 1: Dieses Cmdlet erstellt einen neuen Amazon ECS-Cluster.

```
New-ECSCluster -ClusterName "LAB-ECS-CL" -Setting @{Name="containerInsights";
 Value="enabled"}
```

Ausgabe:

```
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                             : {}
```

- Einzelheiten zur API finden Sie unter CreateCluster AWS Tools for PowerShellCmdlet-Referenz.

`New-ECSService`

Das folgende Codebeispiel zeigt die Verwendung. `New-ECSService`

Tools für PowerShell

Beispiel 1: Dieser Beispielbefehl erstellt in Ihrem Standardcluster einen Dienst namens `ecs-simple-service`. Der Dienst verwendet die Aufgabendefinition `ecs-demo` und verwaltet 10 Instanziierungen dieser Aufgabe.

```
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10
```

Beispiel 2: Dieser Beispielbefehl erstellt einen Dienst hinter einem Load Balancer in Ihrem Standardcluster namens ``. ecs-simple-service Der Dienst verwendet die Aufgabendefinition `ecs-demo` und verwaltet 10 Instanziierungen dieser Aufgabe.

```
$lb = @{
    LoadBalancerName = "EC2Contai-EcsElast-S06278JGSJCM"
    ContainerName = "simple-demo"
    ContainerPort = 80
}
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10 -LoadBalancer $lb
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz. CreateServiceAWS Tools for PowerShell

`Remove-ECSCluster`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ECSCluster`

Tools für PowerShell

Beispiel 1: Dieses Cmdlet löscht den angegebenen ECS-Cluster. Sie müssen alle Container-Instances aus diesem Cluster deregistrieren, bevor Sie ihn löschen können.

```
Remove-ECSCluster -Cluster "LAB-ECS"
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DeleteCluster AWS Tools for PowerShellCmdlet-Referenz.

`Remove-ECSService`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ECSService`

Tools für PowerShell

Beispiel 1: Löscht den Dienst mit dem Namen my-http-service " im Standardcluster. Der Dienst muss die gewünschte Anzahl und die laufende Anzahl 0 haben, bevor Sie ihn löschen können. Sie werden zur Bestätigung aufgefordert, bevor der Befehl ausgeführt wird. Um die Bestätigungsaufforderung zu umgehen, fügen Sie den Schalter -Force hinzu.

```
Remove-ECSService -Service my-http-service
```

Beispiel 2: Löscht den Dienst mit dem Namen 'my-http-service' im benannten Cluster.

```
Remove-ECSService -Cluster myCluster -Service my-http-service
```

- Einzelheiten zur API finden Sie unter DeleteService AWS Tools for PowerShellCmdlet-Referenz.

`Update-ECSClusterSetting`

Das folgende Codebeispiel zeigt die Verwendung. `Update-ECSClusterSetting`

Tools für PowerShell

Beispiel 1: Dieses Cmdlet ändert die Einstellungen, die für einen ECS-Cluster verwendet werden sollen.

```
Update-ECSClusterSetting -Cluster "LAB-ECS-CL" -Setting @{Name="containerInsights";
 Value="disabled"}
```

Ausgabe:

```
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                              : {}
```

- Einzelheiten zur API finden Sie unter UpdateClusterSettingsCmdlet-Referenz.AWS Tools for PowerShell

`Update-ECSService`

Das folgende Codebeispiel zeigt die Verwendung. `Update-ECSService`

Tools für PowerShell

Beispiel 1: Dieser Beispielbefehl aktualisiert den Dienst `my-http-service` so, dass er die Aufgabendefinition amazon-ecs-sample `` verwendet.

```
Update-ECSService -Service my-http-service -TaskDefinition amazon-ecs-sample
```

Beispiel 2: Mit diesem Beispielbefehl wird die gewünschte Anzahl des my-http-service ``-Dienstes auf 10 aktualisiert.

```
Update-ECSService -Service my-http-service -DesiredCount 10
```

- Einzelheiten zur API finden Sie unter UpdateService AWS Tools for PowerShellCmdlet-Referenz.

Amazon EFS-Beispiele für die Verwendung von Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon EFS Aktionen ausführen und allgemeine Szenarien implementieren. AWS Tools for PowerShell

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

Aktionen

`Edit-EFSMountTargetSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung`Edit-EFSMountTargetSecurityGroup`.

Tools für PowerShell

Beispiel 1: Aktualisiert die für das angegebene Mount-Ziel geltenden Sicherheitsgruppen. Bis zu 5 können im Format „sg-xxxxxxxx“ angegeben werden.

```
Edit-EFSMountTargetSecurityGroup -MountTargetId fsmt-1a2b3c4d -SecurityGroup sg-
group1,sg-group3
```

- Einzelheiten zur API finden Sie unter ModifyMountTargetSecurityGroupsCmdlet-Referenz.AWS Tools for PowerShell

`Get-EFSFileSystem`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EFSFileSystem`

Beispiel 1: Gibt die Sammlung aller Dateisysteme zurück, die dem Konto des Anrufers in der Region gehören.

```
Get-EFSFileSystem
```

Ausgabe:

```
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
...
```

Beispiel 2: Gibt die Details des angegebenen Dateisystems zurück.

```
Get-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

Beispiel 3: Gibt die Details eines Dateisystems unter Verwendung des Tokens zur Erstellung der Idempotenz zurück, das bei der Erstellung des Dateisystems angegeben wurde.

```
Get-EFSFileSystem -CreationToken 1a2bff54-85e0-4747-bd95-7bc172c4f555
```

- Einzelheiten zur API finden Sie unter DescribeFileSystems AWS Tools for PowerShellCmdlet-Referenz.

`Get-EFSMountTarget`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EFSMountTarget`

Tools für PowerShell

Beispiel 1: Gibt die Sammlung von Mount-Zielen zurück, die dem angegebenen Dateisystem zugeordnet sind.

```
Get-EFSMountTarget -FileSystemId fs-1a2b3c4d
```

Ausgabe:

```
FileSystemId       : fs-1a2b3c4d
IpAddress          : 10.0.0.131
LifeCycleState     : available
MountTargetId      : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId            : 123456789012
SubnetId           : subnet-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter DescribeMountTargets AWS Tools for PowerShellCmdlet-Referenz.

`Get-EFSMountTargetSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EFSMountTargetSecurityGroup`

Tools für PowerShell

Beispiel 1: Gibt die IDs der Sicherheitsgruppen zurück, die derzeit der Netzwerkschnittstelle zugewiesen sind, die dem Mount-Ziel zugeordnet ist.

```
Get-EFSMountTargetSecurityGroup -MountTargetId fsmt-1a2b3c4d
```

Ausgabe:

```
sg-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter DescribeMountTargetSecurityGroups AWS Tools for PowerShellCmdlet-Referenz.

`Get-EFSTag`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EFSTag`

Tools für PowerShell

Beispiel 1: Gibt die Sammlung von Tags zurück, die derzeit dem angegebenen Dateisystem zugeordnet sind.

```
Get-EFSTag -FileSystemId fs-1a2b3c4d
```

Ausgabe:

```
Key         Value
---         -----
Name        My File System
tagkey1     tagvalue1
tagkey2     tagvalue2
```

- Einzelheiten zur API finden Sie unter DescribeTags AWS Tools for PowerShellCmdlet-Referenz.

`New-EFSFileSystem`

Das folgende Codebeispiel zeigt die Verwendung. `New-EFSFileSystem`

Tools für PowerShell

Beispiel 1: Erzeugt ein neues, leeres Dateisystem. Das Token, das verwendet wird, um sicherzustellen, dass die idempotente Erstellung sichergestellt wird, wird automatisch generiert und kann vom **`CreationToken`** Mitglied des zurückgegebenen Objekts aus aufgerufen werden.

```
New-EFSFileSystem
```

Ausgabe:

```
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
```

Beispiel 2: Erstellt ein neues, leeres Dateisystem unter Verwendung eines benutzerdefinierten Tokens, um eine idempotente Erstellung sicherzustellen.

```
New-EFSFileSystem -CreationToken "MyUniqueToken"
```

- Einzelheiten zur API finden Sie unter CreateFileSystem AWS Tools for PowerShellCmdlet-Referenz.

`New-EFSMountTarget`

Das folgende Codebeispiel zeigt die Verwendung. `New-EFSMountTarget`

Tools für PowerShell

Beispiel 1: Erzeugt ein neues Mount-Ziel für ein Dateisystem. Das angegebene Subnetz wird verwendet, um die Virtual Private Cloud (VPC) zu bestimmen, in der das Mount-Ziel erstellt wird, und die IP-Adresse, die automatisch zugewiesen wird (aus dem Adressbereich des Subnetzes). Die zugewiesene IP-Adresse kann verwendet werden, um dieses Dateisystem dann auf einer Amazon EC2 EC2-Instance zu mounten. Da keine Sicherheitsgruppen angegeben wurden, ist die für das Ziel erstellte Netzwerkschnittstelle der Standardsicherheitsgruppe für die VPC des Subnetzes zugeordnet.

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d
```

Ausgabe:

```
FileSystemId       : fs-1a2b3c4d
IpAddress          : 10.0.0.131
LifeCycleState     : creating
MountTargetId      : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId            : 123456789012
SubnetId           : subnet-1a2b3c4d
```

Beispiel 2: Erzeugt ein neues Mount-Ziel für das angegebene Dateisystem mit automatisch zugewiesener IP-Adresse. Die für das Mount-Ziel erstellte Netzwerkschnittstelle ist den

angegebenen Sicherheitsgruppen zugeordnet (es können bis zu 5 im Format „sg-xxxxxxxx“ angegeben werden).

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -
SecurityGroup sg-group1,sg-group2,sg-group3
```

Beispiel 3: Erstellt ein neues Mount-Ziel für das angegebene Dateisystem mit der angegebenen IP-Adresse.

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -IpAddress
 10.0.0.131
```

- Einzelheiten zur API finden Sie unter CreateMountTarget AWS Tools for PowerShellCmdlet-Referenz.

`New-EFSTag`

Das folgende Codebeispiel zeigt die Verwendung. `New-EFSTag`

Tools für PowerShell

Beispiel 1: Wendet die Sammlung von Tags auf das angegebene Dateisystem an. Wenn ein Tag mit dem angegebenen Schlüssel bereits im Dateisystem vorhanden ist, wird der Wert des Tags aktualisiert.

```
New-EFSTag -FileSystemId fs-1a2b3c4d -Tag
 @{Key="tagkey1";Value="tagvalue1"},@{Key="tagkey2";Value="tagvalue2"}
```

Beispiel 2: Legt das Namens-Tag für das angegebene Dateisystem fest. Dieser Wert wird zusammen mit anderen Dateisystemdetails zurückgegeben, wenn das FileSystem Cmdlet Get-EFS verwendet wird.

```
New-EFSTag -FileSystemId fs-1a2b3c4d -Tag @{Key="Name";Value="My File System"}
```

- Einzelheiten zur API finden Sie unter CreateTagsCmdlet-Referenz.AWS Tools for PowerShell

`Remove-EFSFileSystem`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EFSFileSystem`

Tools für PowerShell

Beispiel 1: Löscht das angegebene Dateisystem, das nicht mehr verwendet wird (wenn das Dateisystem Mount-Ziele hat, müssen diese zuerst entfernt werden). Sie werden zur Bestätigung aufgefordert, bevor das Cmdlet fortfährt. Verwenden Sie die Befehlszeilenoption, um die Bestätigung zu unterdrücken. **`-Force`**

```
Remove-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter DeleteFileSystem AWS Tools for PowerShellCmdlet-Referenz.

`Remove-EFSMountTarget`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EFSMountTarget`

Tools für PowerShell

Beispiel 1: Löscht das angegebene Mount-Ziel. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird. Verwenden Sie den **`-Force`** Schalter, um die Aufforderung zu unterdrücken. Beachten Sie, dass durch diesen Vorgang alle Einhängungen des Dateisystems über das Ziel unterbrochen werden. Falls möglich, sollten Sie in Erwägung ziehen, das Dateisystem vor der Ausführung dieses Befehls aufzuheben.

```
Remove-EFSMountTarget -MountTargetId fsmt-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter DeleteMountTargetCmdlet-Referenz.AWS Tools for PowerShell

`Remove-EFSTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EFSTag`

Tools für PowerShell

Beispiel 1: Löscht die Sammlung eines oder mehrerer Tags aus einem Dateisystem. Sie werden zur Bestätigung aufgefordert, bevor das Cmdlet fortfährt. Verwenden Sie die Befehlszeilenoption, um die Bestätigung zu unterdrücken. **`-Force`**

```
Remove-EFSTag -FileSystemId fs-1a2b3c4d -TagKey "tagkey1","tagkey2"
```

- Einzelheiten zur API finden Sie unter DeleteTags AWS Tools for PowerShellCmdlet-Referenz.

Amazon EKS-Beispiele für die Verwendung von Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS Tools for PowerShell mit Amazon EKS Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`Add-EKSResourceTag`

Das folgende Codebeispiel zeigt die Verwendung`Add-EKSResourceTag`.

Tools für PowerShell

Beispiel 1: Dieses Cmdlet ordnet die angegebenen Tags einer Ressource mit dem angegebenen ResourceArn zu.

```
Add-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD" -
Tag @{Name = "EKSPRODCLUSTER"}
```

- Einzelheiten zur API finden Sie unter TagResourceCmdlet-Referenz.AWS Tools for PowerShell

`Get-EKSCluster`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EKSCluster`

Tools für PowerShell

Beispiel 1: Dieses Cmdlet gibt beschreibende Informationen zu einem Amazon EKS-Cluster zurück.

```
Get-EKSCluster -Name "PROD"
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeClusterCmdlet-Referenz.AWS Tools for PowerShell

`Get-EKSClusterList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EKSClusterList`

Tools für PowerShell

Beispiel 1: Dieses Cmdlet listet die Amazon EKS-Cluster in Ihrem AWS-Konto in der angegebenen Region auf.

```
Get-EKSClusterList
```

Ausgabe:

```
PROD
```

- Einzelheiten zur API finden Sie unter ListClusters AWS Tools for PowerShellCmdlet-Referenz.

`Get-EKSFargateProfile`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EKSFargateProfile`

Tools für PowerShell

Beispiel 1: Dieses Cmdlet gibt beschreibende Informationen zu einem AWS Fargate-Profil zurück.

```
Get-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

Ausgabe:

```
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                : {}
```

- Einzelheiten zur API finden Sie unter DescribeFargateProfileCmdlet-Referenz.AWS Tools for PowerShell

`Get-EKSFargateProfileList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EKSFargateProfileList`

Tools für PowerShell

Beispiel 1: Dieses Cmdlet listet die AWS Fargate-Profile auf, die dem angegebenen Cluster in Ihrem AWS-Konto in der angegebenen Region zugeordnet sind.

```
Get-EKSFargateProfileList -ClusterName "TEST"
```

Ausgabe:

```
EKSFargate
EKSFargateProfile
```

- Einzelheiten zur API finden Sie unter ListFargateProfilesCmdlet-Referenz.AWS Tools for PowerShell

`Get-EKSNodegroup`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EKSNodegroup`

Tools für PowerShell

Beispiel 1: Dieses Cmdlet gibt beschreibende Informationen zu einer Amazon EKS-Knotengruppe zurück.

```
Get-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeNodegroupCmdlet-Referenz.AWS Tools for PowerShell

`Get-EKSNodegroupList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EKSNodegroupList`

Tools für PowerShell

Beispiel 1: Dieses Cmdlet listet die Amazon EKS-Knotengruppen auf, die dem angegebenen Cluster in Ihrem AWS-Konto in der angegebenen Region zugeordnet sind.

```
Get-EKSNodegroupList -ClusterName PROD
```

Ausgabe:

```
ProdEKSNodeGroup
```

- Einzelheiten zur API finden Sie unter ListNodegroups AWS Tools for PowerShellCmdlet-Referenz.

`Get-EKSResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EKSResourceTag`

Tools für PowerShell

Beispiel 1: Dieses Cmdlet listet die Tags für eine Amazon EKS-Ressource auf.

```
Get-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
```

Ausgabe:

```
Key  Value
---  -----
Name EKSPRODCLUSTER
```

- Einzelheiten zur API finden Sie unter ListTagsForResource AWS Tools for PowerShellCmdlet-Referenz.

`Get-EKSUpdate`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EKSUpdate`

Tools für PowerShell

Beispiel 1: Dieses Cmdlet gibt beschreibende Informationen zu einem Update für Ihren Amazon EKS-Cluster oder die zugehörige verwaltete Knotengruppe zurück.

```
Get-EKSUpdate -Name "PROD" -UpdateId "ee708232-7d2e-4ed7-9270-d0b5176f0726"
```

Ausgabe:

```
CreatedAt : 12/25/2019 5:03:07 PM
Errors    : {}
Id        : ee708232-7d2e-4ed7-9270-d0b5176f0726
Params    : {Amazon.EKS.Model.UpdateParam}
Status    : Successful
Type      : LoggingUpdate
```

- Einzelheiten zur API finden Sie unter DescribeUpdateCmdlet-Referenz.AWS Tools for PowerShell

`Get-EKSUpdateList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EKSUpdateList`

Tools für PowerShell

Beispiel 1: Dieses Cmdlet listet die Updates auf, die einem Amazon EKS-Cluster oder einer verwalteten Knotengruppe in Ihrer AWS-Konto Region zugeordnet sind.

```
Get-EKSUpdateList -Name "PROD"
```

Ausgabe:

```
ee708232-7d2e-4ed7-9270-d0b5176f0726
```

- Einzelheiten zur API finden Sie unter ListUpdates AWS Tools for PowerShellCmdlet-Referenz.

`New-EKSCluster`

Das folgende Codebeispiel zeigt die Verwendung. `New-EKSCluster`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein neuer Cluster namens 'prod' erstellt.

```
New-EKSCluster -Name prod -ResourcesVpcConfig
 @{SubnetIds=@("subnet-0a1b2c3d","subnet-3a2b1c0d");SecurityGroupIds="sg-6979fe18"}
 -RoleArn "arn:aws:iam::012345678901:role/eks-service-role"
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter CreateCluster AWS Tools for PowerShellCmdlet-Referenz.

`New-EKSFargateProfile`

Das folgende Codebeispiel zeigt die Verwendung. `New-EKSFargateProfile`

Tools für PowerShell

Beispiel 1: Dieses Cmdlet erstellt ein AWS Fargate-Profil für Ihren Amazon EKS-Cluster. Sie müssen mindestens ein Fargate-Profil in einem Cluster haben, um Pods auf der Fargate-Infrastruktur planen zu können.

```
New-EKSFargateProfile -FargateProfileName EKSFargateProfile -ClusterName TEST -
Subnet "subnet-02f6ff500ff2067a0", "subnet-0cd976f08d5fbfaae" -PodExecutionRoleArn
 arn:aws:iam::012345678912:role/AmazonEKSFargatePodExecutionRole -Selector
 @{Namespace="default"}
```

Ausgabe:

```
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                : {}
```

- Einzelheiten zur API finden Sie unter CreateFargateProfile AWS Tools for PowerShellCmdlet-Referenz.

`New-EKSNodeGroup`

Das folgende Codebeispiel zeigt die Verwendung. `New-EKSNodeGroup`

Tools für PowerShell

Beispiel 1: Dieses Cmdlet erstellt eine verwaltete Worker-Knotengruppe für einen Amazon EKS-Cluster. Sie können nur eine Knotengruppe für Ihren Cluster erstellen, die der aktuellen Kubernetes-Version für den Cluster entspricht. Alle Knotengruppen werden mit der neuesten AMI-Release-Version für die jeweilige Kubernetes-Unterversion des Clusters erstellt.

```
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"
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter CreateNodegroup AWS Tools for PowerShellCmdlet-Referenz.

`Remove-EKSCluster`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EKSCluster`

Tools für PowerShell

Beispiel 1: Dieses Cmdlet löscht die Amazon EKS-Cluster-Steuerebene.

```
Remove-EKSCluster -Name "DEV-KUBE-CL"
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DeleteClusterCmdlet-Referenz.AWS Tools for PowerShell

`Remove-EKSFargateProfile`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EKSFargateProfile`

Tools für PowerShell

Beispiel 1: Dieses Cmdlet löscht ein AWS Fargate-Profil. Wenn Sie ein Fargate-Profil löschen, werden alle Pods, die auf Fargate laufen und mit dem Profil erstellt wurden, gelöscht.

```
Remove-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

Ausgabe:

```
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                : {}
```

- Einzelheiten zur API finden Sie unter DeleteFargateProfile AWS Tools for PowerShellCmdlet-Referenz.

`Remove-EKSNodegroup`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EKSNodegroup`

Tools für PowerShell

Beispiel 1: Dieses Cmdlet löscht eine Amazon EKS-Knotengruppe für einen Cluster.

```
Remove-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DeleteNodegroupCmdlet-Referenz.AWS Tools for PowerShell

`Remove-EKSResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EKSResourceTag`

Tools für PowerShell

Beispiel 1: Dieses Cmdlet löscht angegebene Tags aus einer EKS-Ressource.

```
Remove-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
 -TagKey "Name"
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter UntagResourceCmdlet-Referenz.AWS Tools for PowerShell

`Update-EKSClusterConfig`

Das folgende Codebeispiel zeigt die Verwendung. `Update-EKSClusterConfig`

Tools für PowerShell

Beispiel 1: Aktualisiert eine Amazon EKS-Clusterkonfiguration. Ihr Cluster funktioniert während des Updates weiterhin.

```
Update-EKSClusterConfig -Name "PROD" -Logging_ClusterLogging
 @{Types="api","audit","authenticator","controllerManager","scheduler",Enabled="True"}
```

Ausgabe:

```
CreatedAt : 12/25/2019 5:03:07 PM
Errors    : {}
Id        : ee708232-7d2e-4ed7-9270-d0b5176f0726
Params    : {Amazon.EKS.Model.UpdateParam}
Status    : InProgress
Type      : LoggingUpdate
```

- Einzelheiten zur API finden Sie unter UpdateClusterConfig AWS Tools for PowerShellCmdlet-Referenz.

`Update-EKSClusterVersion`

Das folgende Codebeispiel zeigt die Verwendung. `Update-EKSClusterVersion`

Tools für PowerShell

Beispiel 1: Dieses Cmdlet aktualisiert einen Amazon EKS-Cluster auf die angegebene Kubernetes-Version. Ihr Cluster funktioniert während des Updates weiterhin.

```
Update-EKSClusterVersion -Name "PROD-KUBE-CL" -Version 1.14
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter UpdateClusterVersion AWS Tools for PowerShellCmdlet-Referenz.

Elastic Load Balancing — Beispiele für Version 1 mit Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS Tools for PowerShell mit Elastic Load Balancing — Version 1 Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zur Einrichtung und Ausführung des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`Add-ELBLoadBalancerToSubnet`

Das folgende Codebeispiel zeigt die VerwendungAdd-ELBLoadBalancerToSubnet.

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Subnetz der Gruppe von Subnetzen hinzugefügt, die für den angegebenen Load Balancer konfiguriert sind. Die Ausgabe enthält die vollständige Liste der Subnetze.

```
Add-ELBLoadBalancerToSubnet -LoadBalancerName my-load-balancer -Subnet
 subnet-12345678
```

Ausgabe:

```
subnet-12345678
subnet-87654321
```

- Einzelheiten zur API finden Sie unter AttachLoadBalancerToSubnets AWS Tools for PowerShellCmdlet-Referenz.

`Add-ELBResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Add-ELBResourceTag`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die angegebenen Tags zum angegebenen Load Balancer hinzugefügt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag
 @{ Key="project";Value="lima" },@{ Key="department";Value="digital-media" }
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um ein Tag für den Tag-Parameter zu erstellen.

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.Tag
$tag.Key = "project"
$tag.Value = "lima"
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag $tag
```

- Einzelheiten zur API finden Sie unter AddTags AWS Tools for PowerShellCmdlet-Referenz.

`Disable-ELBAvailabilityZoneForLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung. `Disable-ELBAvailabilityZoneForLoadBalancer`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Availability Zone aus dem angegebenen Load Balancer entfernt. Die Ausgabe umfasst die verbleibenden Availability Zones.

```
Disable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -
AvailabilityZone us-west-2a
```

Ausgabe:

```
us-west-2b
```

- Einzelheiten zur API finden Sie unter DisableAvailabilityZonesForLoadBalancer AWS Tools for PowerShellCmdlet-Referenz.

`Dismount-ELBLoadBalancerFromSubnet`

Das folgende Codebeispiel zeigt die Verwendung. `Dismount-ELBLoadBalancerFromSubnet`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Subnetz aus der Gruppe von Subnetzen entfernt, die für den angegebenen Load Balancer konfiguriert sind. Die Ausgabe umfasst die verbleibenden Subnetze.

```
Dismount-ELBLoadBalancerFromSubnet -LoadBalancerName my-load-balancer -Subnet
 subnet-12345678
```

Ausgabe:

```
subnet-87654321
```

- Einzelheiten zur API finden Sie unter DetachLoadBalancerFromSubnets AWS Tools for PowerShellCmdlet-Referenz.

`Edit-ELBLoadBalancerAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-ELBLoadBalancerAttribute`

Tools für PowerShell

Beispiel 1: Dieses Beispiel aktiviert den zonenübergreifenden Load Balancer für den angegebenen Load Balancer.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -
CrossZoneLoadBalancing_Enabled $true
```

Beispiel 2: In diesem Beispiel wird der Verbindungsabbau für den angegebenen Load Balancer deaktiviert.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -
ConnectionDraining_Enabled $false
```

Beispiel 3: In diesem Beispiel wird die Zugriffsprotokollierung für den angegebenen Load Balancer aktiviert.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer `
>> -AccessLog_Enabled $true `
>> -AccessLog_S3BucketName my-logs-bucket `
>> -AccessLog_S3BucketPrefix my-app/prod `
>> -AccessLog_EmitInterval 60
```

- Einzelheiten zur API finden Sie unter ModifyLoadBalancerAttributes AWS Tools for PowerShellCmdlet-Referenz.

`Enable-ELBAvailabilityZoneForLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung. `Enable-ELBAvailabilityZoneForLoadBalancer`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Availability Zone zum angegebenen Load Balancer hinzugefügt. Die Ausgabe enthält die vollständige Liste der Availability Zones.

```
Enable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -
AvailabilityZone us-west-2a
```

Ausgabe:

```
us-west-2a
us-west-2b
```

- Einzelheiten zur API finden Sie unter [EnableAvailabilityZonesForLoadBalancer AWS Tools for PowerShell](#)Cmdlet-Referenz.

`Get-ELBInstanceHealth`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELBInstanceHealth`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt den Status der Instances, die beim angegebenen Load Balancer registriert sind.

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer
```

Ausgabe:

```
Description                  InstanceId                   ReasonCode
      State
-----------                  ----------                   ----------
      -----
N/A                          i-87654321                   N/A
      InService
Instance has failed at lea... i-12345678                  Instance
      OutOfService
```

Beispiel 2: Dieses Beispiel beschreibt den Status der angegebenen Instance, die beim angegebenen Load Balancer registriert ist.

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance i-12345678
```

Beispiel 3: In diesem Beispiel wird die vollständige Beschreibung des Status der angegebenen Instanz angezeigt.

```
(Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance
 i-12345678).Description
```

Ausgabe:

```
Instance has failed at least the UnhealthyThreshold number of health checks
 consecutively.
```

- Einzelheiten zur API finden Sie unter DescribeInstanceHealth AWS Tools for PowerShellCmdlet-Referenz.

`Get-ELBLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELBLoadBalancer`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Namen Ihrer Load Balancer aufgeführt.

```
Get-ELBLoadBalancer | format-table -property LoadBalancerName
```

Ausgabe:

```
LoadBalancerName
----------------
my-load-balancer
my-other-load-balancer
my-internal-load-balancer
```

Beispiel 2: Dieses Beispiel beschreibt den angegebenen Load Balancer.

```
Get-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

Ausgabe:

```
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
```

Beispiel 3: Dieses Beispiel beschreibt alle Ihre Load Balancer in der aktuellen AWS Region.

```
Get-ELBLoadBalancer
```

Beispiel 4: In diesem Beispiel werden alle Ihre Load Balancer für alle verfügbaren Load Balancer beschrieben. AWS-Regionen

```
Get-AWSRegion | % { Get-ELBLoadBalancer -Region $_ }
```

- Einzelheiten zur API finden Sie unter DescribeLoadBalancers AWS Tools for PowerShellCmdlet-Referenz.

`Get-ELBLoadBalancerAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELBLoadBalancerAttribute`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Attribute für den angegebenen Load Balancer beschrieben.

```
Get-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer
```

Ausgabe:

```
AccessLog              : Amazon.ElasticLoadBalancing.Model.AccessLog
AdditionalAttributes   : {}
ConnectionDraining     : Amazon.ElasticLoadBalancing.Model.ConnectionDraining
ConnectionSettings     : Amazon.ElasticLoadBalancing.Model.ConnectionSettings
CrossZoneLoadBalancing : Amazon.ElasticLoadBalancing.Model.CrossZoneLoadBalancing
```

- Einzelheiten zur API finden Sie unter DescribeLoadBalancerAttributes AWS Tools for PowerShellCmdlet-Referenz.

`Get-ELBLoadBalancerPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELBLoadBalancerPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Richtlinien beschrieben, die dem angegebenen Load Balancer zugeordnet sind.

```
Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer
```

Ausgabe:

```
PolicyAttributeDescriptions             PolicyName
 PolicyTypeName
---------------------------             ----------
 --------------
{ProxyProtocol}                         my-ProxyProtocol-policy
 ProxyProtocolPolicyType
{CookieName}                            my-app-cookie-policy
 AppCookieStickinessPolicyType
```

Beispiel 2: In diesem Beispiel werden die Attribute der angegebenen Richtlinie beschrieben.

```
(Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
ProxyProtocol-policy).PolicyAttributeDescriptions
```

Ausgabe:

```
AttributeName    AttributeValue
-------------    --------------
ProxyProtocol    true
```

Beispiel 3: Dieses Beispiel beschreibt die vordefinierten Richtlinien, einschließlich der Beispielrichtlinien. Die Namen der Beispielrichtlinien haben das Präfix ElbSample-.

```
Get-ELBLoadBalancerPolicy
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter [DescribeLoadBalancerPolicies AWS Tools for PowerShell](#)Cmdlet-Referenz.

`Get-ELBLoadBalancerPolicyType`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELBLoadBalancerPolicyType`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Richtlinientypen abgerufen, die von Elastic Load Balancing unterstützt werden.

```
Get-ELBLoadBalancerPolicyType
```

Ausgabe:

```
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
```

Beispiel 2: Dieses Beispiel beschreibt den angegebenen Richtlinientyp.

```
Get-ELBLoadBalancerPolicyType -PolicyTypeName ProxyProtocolPolicyType
```

Ausgabe:

```
Description                             PolicyAttributeTypeDescriptions
 PolicyTypeName
-----------                             -------------------------------
 --------------
Policy that controls whether to incl... {ProxyProtocol}
 ProxyProtocolPolicyType
```

Beispiel 3: In diesem Beispiel wird die vollständige Beschreibung des angegebenen Richtlinientyps angezeigt.

```
(Get-ELBLoadBalancerPolicyType -PolicyTypeName).Description
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter [DescribeLoadBalancerPolicyTypes AWS Tools for PowerShell](Cmdlet-Referenz).

`Get-ELBResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELBResourceTag`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Tags für die angegebenen Load Balancer aufgelistet.

```
Get-ELBResourceTag -LoadBalancerName @("my-load-balancer","my-internal-load-
balancer")
```

Ausgabe:

```
LoadBalancerName             Tags
----------------             ----
my-load-balancer             {project, department}
my-internal-load-balancer    {project, department}
```

Beispiel 2: In diesem Beispiel werden die Tags für den angegebenen Load Balancer beschrieben.

```
(Get-ELBResourceTag -LoadBalancerName my-load-balancer).Tags
```

Ausgabe:

```
Key           Value
---           -----
project       lima
department    digital-media
```

- Einzelheiten zur API finden Sie unter DescribeTags AWS Tools for PowerShellCmdlet-Referenz.

`Join-ELBSecurityGroupToLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung. `Join-ELBSecurityGroupToLoadBalancer`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die aktuelle Sicherheitsgruppe für den angegebenen Load Balancer durch die angegebene Sicherheitsgruppe ersetzt.

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -
SecurityGroup sg-87654321
```

Ausgabe:

```
sg-87654321
```

Beispiel 2: Um die aktuelle Sicherheitsgruppe beizubehalten und eine zusätzliche Sicherheitsgruppe anzugeben, geben Sie sowohl die vorhandene als auch die neue Sicherheitsgruppe an.

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -
SecurityGroup @("sg-12345678", "sg-87654321")
```

Ausgabe:

```
sg-12345678
sg-87654321
```

- Einzelheiten zur API finden Sie unter ApplySecurityGroupsToLoadBalancer AWS Tools for PowerShellCmdlet-Referenz.

`New-ELBAppCookieStickinessPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `New-ELBAppCookieStickinessPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Stickiness-Richtlinie erstellt, die sich an die Lebensdauer des angegebenen, von der Anwendung generierten Cookies für Sticky-Sitzungen hält.

```
New-ELBAppCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-
app-cookie-policy -CookieName my-app-cookie
```

- Einzelheiten zur API finden Sie unter CreateAppCookieStickinessPolicyCmdlet-Referenz.AWS Tools for PowerShell

`New-ELBLBCookieStickinessPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `New-ELBLBCookieStickinessPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Sperrrichtlinie erstellt, bei der die Lebensdauer von Sperrsitzungen durch den angegebenen Ablaufzeitraum (in Sekunden) gesteuert wird.

```
New-ELBLBCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-
duration-cookie-policy -CookieExpirationPeriod 60
```

Beispiel 2: In diesem Beispiel wird eine Stickiness-Richtlinie erstellt, bei der die Lebensdauer von Sperrsitzungen durch die Lebensdauer des Browsers (User-Agent) gesteuert wird.

```
New-ELBLBCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-
duration-cookie-policy
```

- Einzelheiten zur API finden Sie unter CreateLbCookieStickinessPolicyCmdlet-Referenz.AWS Tools for PowerShell

`New-ELBLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung. `New-ELBLoadBalancer`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Load Balancer mit einem HTTP-Listener in einer VPC erstellt.

```
$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
```

Beispiel 2: In diesem Beispiel wird ein Load Balancer mit einem HTTP-Listener in EC2-Classic erstellt.

```
New-ELBLoadBalancer -LoadBalancerName my-classic-load-balancer -AvailabilityZone us-
west-2a` -Listener $httpListener
```

Ausgabe:

```
my-classic-load-balancer-123456789.us-west-2.elb.amazonaws.com
```

Beispiel 3: In diesem Beispiel wird ein Load Balancer mit einem HTTPS-Listener erstellt.

```
$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
```

- Einzelheiten zur API finden Sie unter CreateLoadBalancer AWS Tools for PowerShellCmdlet-Referenz.

`New-ELBLoadBalancerListener`

Das folgende Codebeispiel zeigt die Verwendung. `New-ELBLoadBalancerListener`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird dem angegebenen Load Balancer ein HTTPS-Listener hinzugefügt.

```
$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
```

- Einzelheiten zur API finden Sie unter CreateLoadBalancerListeners AWS Tools for PowerShellCmdlet-Referenz.

`New-ELBLoadBalancerPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `New-ELBLoadBalancerPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine neue Proxy-Protokollrichtlinie für einen angegebenen Load Balancer erstellt.

```
$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
```

- Einzelheiten zur API finden Sie unter CreateLoadBalancerPolicy AWS Tools for PowerShellCmdlet-Referenz.

`Register-ELBInstanceWithLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung. `Register-ELBInstanceWithLoadBalancer`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene EC2-Instance beim angegebenen Load Balancer registriert.

```
Register-ELBInstanceWithLoadBalancer -LoadBalancerName my-load-balancer -Instance
 i-12345678
```

Ausgabe:

```
InstanceId
----------
i-12345678
i-87654321
```

- Einzelheiten zur API finden Sie unter RegisterInstancesWithLoadBalancer AWS Tools for PowerShellCmdlet-Referenz.

`Remove-ELBInstanceFromLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ELBInstanceFromLoadBalancer`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene EC2-Instance aus dem angegebenen Load Balancer entfernt. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-ELBInstanceFromLoadBalancer -LoadBalancerName my-load-balancer -Instance
 i-12345678
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DeregisterInstancesFromLoadBalancer AWS Tools for PowerShellCmdlet-Referenz.

`Remove-ELBLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ELBLoadBalancer`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Load Balancer gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

Ausgabe:

```
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"):
```

- Einzelheiten zur API finden Sie unter DeleteLoadBalancer AWS Tools for PowerShellCmdlet-Referenz.

`Remove-ELBLoadBalancerListener`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ELBLoadBalancerListener`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der Listener auf Port 80 für den angegebenen Load Balancer gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-ELBLoadBalancerListener -LoadBalancerName my-load-balancer -LoadBalancerPort
 80
```

Ausgabe:

```
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"):
```

- Einzelheiten zur API finden Sie unter DeleteLoadBalancerListeners AWS Tools for PowerShellCmdlet-Referenz.

`Remove-ELBLoadBalancerPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ELBLoadBalancerPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Richtlinie aus dem angegebenen Load Balancer gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
duration-cookie-policy
```

Ausgabe:

```
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"):
```

- Einzelheiten zur API finden Sie unter DeleteLoadBalancerPolicy AWS Tools for PowerShellCmdlet-Referenz.

`Remove-ELBResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ELBResourceTag`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Tag aus dem angegebenen Load Balancer entfernt. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie

nicht auch den Force-Parameter angeben. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Remove-ELBResourceTag -LoadBalancerName my-load-balancer -Tag @{ Key="project" }
```

Ausgabe:

```
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"):
```

Beispiel 2: Bei Powershell Version 2 müssen Sie New-Object verwenden, um das Tag für den Tag-Parameter zu erstellen.

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.TagKeyOnly
$tag.Key = "project"
Remove-ELBResourceTag -Tag $tag -Force
```

- Einzelheiten zur API finden Sie unter RemoveTagsCmdlet-Referenz.AWS Tools for PowerShell

`Set-ELBHealthCheck`

Das folgende Codebeispiel zeigt die Verwendung. `Set-ELBHealthCheck`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Einstellungen für die Integritätsprüfung für den angegebenen Load Balancer konfiguriert.

```
Set-ELBHealthCheck -LoadBalancerName my-load-balancer `
>> -HealthCheck_HealthyThreshold 2 `
>> -HealthCheck_UnhealthyThreshold 2 `
>> -HealthCheck_Target "HTTP:80/ping" `
>> -HealthCheck_Interval 30 `
>> -HealthCheck_Timeout 3
```

Ausgabe:

```
HealthyThreshold   : 2
Interval           : 30
Target             : HTTP:80/ping
Timeout            : 3
UnhealthyThreshold : 2
```

- Einzelheiten zur API finden Sie unter ConfigureHealthCheck AWS Tools for PowerShellCmdlet-Referenz.

`Set-ELBLoadBalancerListenerSSLCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `Set-ELBLoadBalancerListenerSSLCertificate`

Tools für PowerShell

Beispiel 1: Dieses Beispiel ersetzt das Zertifikat, das die SSL-Verbindungen für den angegebenen Listener beendet.

```
Set-ELBLoadBalancerListenerSSLCertificate -LoadBalancerName my-load-balancer `
>> -LoadBalancerPort 443 `
>> -SSLCertificateId "arn:aws:iam::123456789012:server-certificate/new-server-cert"
```

- Einzelheiten zur API finden Sie unter SetLoadBalancerListenerSslCertificate AWS Tools for PowerShellCmdlet-Referenz.

`Set-ELBLoadBalancerPolicyForBackendServer`

Das folgende Codebeispiel zeigt die Verwendung. `Set-ELBLoadBalancerPolicyForBackendServer`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Richtlinien für den angegebenen Port durch die angegebene Richtlinie ersetzt.

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -
InstancePort 80 -PolicyName my-ProxyProtocol-policy
```

Beispiel 2: In diesem Beispiel werden alle Richtlinien entfernt, die dem angegebenen Port zugeordnet sind.

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -
InstancePort 80
```

- Einzelheiten zur API finden Sie unter SetLoadBalancerPoliciesForBackendServer AWS Tools for PowerShellCmdlet-Referenz.

`Set-ELBLoadBalancerPolicyOfListener`

Das folgende Codebeispiel zeigt die Verwendung. `Set-ELBLoadBalancerPolicyOfListener`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Richtlinien für den angegebenen Listener durch die angegebene Richtlinie ersetzt.

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 443 -PolicyName my-SSLNegotiation-policy
```

Beispiel 2: In diesem Beispiel werden alle Richtlinien entfernt, die dem angegebenen Listener zugeordnet sind.

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 443
```

- Einzelheiten zur API finden Sie unter SetLoadBalancerPoliciesOfListener AWS Tools for PowerShellCmdlet-Referenz.

Elastic Load Balancing — Beispiele für Version 2 mit Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS Tools for PowerShell mit Elastic Load Balancing — Version 2 Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zur Einrichtung und Ausführung des Codes im Kontext finden.

Themen

Aktionen

`Add-ELB2ListenerCertificate`

Das folgende Codebeispiel zeigt die Verwendung`Add-ELB2ListenerCertificate`.

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird dem angegebenen Listener ein zusätzliches Zertifikat hinzugefügt.

```
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'}
```

Ausgabe:

```
CertificateArn
 IsDefault
--------------
 ---------
arn:aws:acm:us-east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97
 False
```

- Einzelheiten zur API finden Sie unter AddListenerCertificates AWS Tools for PowerShellCmdlet-Referenz.

`Add-ELB2Tag`

Das folgende Codebeispiel zeigt die Verwendung. `Add-ELB2Tag`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der angegebenen **AWS.Tools.ElasticLoadBalancingV2** Ressource ein neues Tag hinzugefügt.

```
Add-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Tag @{Key =
 'productVersion'; Value = '1.0.0'}
```

- Einzelheiten zur API finden Sie unter AddTags AWS Tools for PowerShellCmdlet-Referenz.

`Edit-ELB2Listener`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-ELB2Listener`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Standardaktion des angegebenen Listeners auf Fixed-Response geändert.

```
$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
```

Ausgabe:

```
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       :
```

- Einzelheiten zur API finden Sie unter ModifyListenerCmdlet-Referenz.AWS Tools for PowerShell

`Edit-ELB2LoadBalancerAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-ELB2LoadBalancerAttribute`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Attribute des angegebenen Load Balancers geändert.

```
Edit-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Attribute @{Key =
 'deletion_protection.enabled'; Value = 'true'}
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter ModifyLoadBalancerAttributes AWS Tools for PowerShellCmdlet-Referenz.

`Edit-ELB2Rule`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-ELB2Rule`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die angegebenen Listener-Regelkonfigurationen geändert.

```
$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
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter ModifyRule AWS Tools for PowerShellCmdlet-Referenz.

`Edit-ELB2TargetGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-ELB2TargetGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Eigenschaften der angegebenen Zielgruppe geändert.

```
Edit-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -HealthCheckIntervalSecond
 60 -HealthCheckPath '/index.html' -HealthCheckPort 8080
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter ModifyTargetGroup AWS Tools for PowerShellCmdlet-Referenz.

`Edit-ELB2TargetGroupAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-ELB2TargetGroupAttribute`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Attribut deregistration_delay der angegebenen Zielgruppe geändert.

```
Edit-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Attribute @{Key =
 'deregistration_delay.timeout_seconds'; Value = 600}
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz. ModifyTargetGroupAttributesAWS Tools for PowerShell

`Get-ELB2AccountLimit`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELB2AccountLimit`

Tools für PowerShell

Beispiel 1: Dieser Befehl listet die ELB2-Kontolimits für eine bestimmte Region auf.

```
Get-ELB2AccountLimit
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeAccountLimits AWS Tools for PowerShellCmdlet-Referenz.

`Get-ELB2Listener`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELB2Listener`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt Listener des angegebenen ALB/NLB.

```
Get-ELB2Listener -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeListenersCmdlet-Referenz.AWS Tools for PowerShell

`Get-ELB2ListenerCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELB2ListenerCertificate`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt das Zertifikat für den angegebenen Listener.

```
Get-ELB2ListenerCertificate -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Ausgabe:

```
CertificateArn
 IsDefault
--------------
 ---------
```

```
arn:aws:acm:us-east-1:123456789012:certificate/5fc7c092-68bf-4862-969c-22fd48b6e17c
 True
```

- Einzelheiten zur API finden Sie unter DescribeListenerCertificates AWS Tools for PowerShellCmdlet-Referenz.

`Get-ELB2LoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELB2LoadBalancer`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden alle Load Balancer für die angegebene Region angezeigt.

```
Get-ELB2LoadBalancer
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeLoadBalancers AWS Tools for PowerShellCmdlet-Referenz.

`Get-ELB2LoadBalancerAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELB2LoadBalancerAttribute`

Tools für PowerShell

Beispiel 1: Dieser Befehl beschreibt die Attribute eines bestimmten Load Balancers.

```
Get-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/net/test-elb/238d34ad8d94bc2e'
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeLoadBalancerAttributes AWS Tools for PowerShellCmdlet-Referenz.

`Get-ELB2Rule`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELB2Rule`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die Listener-Regeln für den angegebenen Listener-ARN.

```
Get-ELB2Rule -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeRules AWS Tools for PowerShellCmdlet-Referenz.

`Get-ELB2SSLPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELB2SSLPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden alle verfügbaren Listener-Richtlinien für ElasticLoadBalancing V2 aufgeführt.

```
Get-ELB2SSLPolicy
```

Ausgabe:

```
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}
```

- Einzelheiten zur API finden Sie unter DescribeSslPolicies AWS Tools for PowerShellCmdlet-Referenz.

`Get-ELB2Tag`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELB2Tag`

Tools für PowerShell

Beispiel 1: Dieses Beispiel listet die Tags für die angegebene Ressource auf.

```
Get-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Ausgabe:

```
ResourceArn
           Tags
-----------
           ----
arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f {stage, internalName, version}
```

- Einzelheiten zur API finden Sie unter DescribeTags AWS Tools for PowerShellCmdlet-Referenz.

`Get-ELB2TargetGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELB2TargetGroup`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die angegebene Zielgruppe.

```
Get-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeTargetGroups AWS Tools for PowerShellCmdlet-Referenz.

`Get-ELB2TargetGroupAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELB2TargetGroupAttribute`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die Attribute der angegebenen Zielgruppe.

```
Get-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeTargetGroupAttributes AWS Tools for PowerShellCmdlet-Referenz.

`Get-ELB2TargetHealth`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELB2TargetHealth`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der Gesundheitsstatus der Ziele zurückgegeben, die in der angegebenen Zielgruppe vorhanden sind.

```
Get-ELB2TargetHealth -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Ausgabe:

```
HealthCheckPort Target                                              TargetHealth
--------------- ------                                              ------------
80              Amazon.ElasticLoadBalancingV2.Model.TargetDescription
 Amazon.ElasticLoadBalancingV2.Model.TargetHealth
```

- Einzelheiten zur API finden Sie unter DescribeTargetHealth AWS Tools for PowerShellCmdlet-Referenz.

`New-ELB2Listener`

Das folgende Codebeispiel zeigt die Verwendung. `New-ELB2Listener`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein neuer ALB-Listener mit der Standardaktion „Forward“ erstellt, um Traffic an die angegebene Zielgruppe zu senden.

```
$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
```

Ausgabe:

```
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       :
```

- Einzelheiten zur API finden Sie unter CreateListener AWS Tools for PowerShellCmdlet-Referenz.

`New-ELB2LoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung. `New-ELB2LoadBalancer`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein neuer, mit dem Internet verbundener Application Load Balancer mit zwei Subnetzen erstellt.

```
New-ELB2LoadBalancer -Type application -Scheme internet-facing -IpAddressType
 ipv4 -Name 'New-Test-ALB' -SecurityGroup 'sg-07c3414abb8811cbd' -subnet 'subnet-
c37a67a6','subnet-fc02eea0'
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter CreateLoadBalancer AWS Tools for PowerShellCmdlet-Referenz.

`New-ELB2Rule`

Das folgende Codebeispiel zeigt die Verwendung. `New-ELB2Rule`

Beispiel 1: In diesem Beispiel wird eine neue Listener-Regel mit einer Aktion mit fester Antwort erstellt, die auf dem Kunden-Header-Wert für den angegebenen Listener basiert.

```
$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
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter CreateRuleCmdlet-Referenz.AWS Tools for PowerShell

`New-ELB2TargetGroup`

Das folgende Codebeispiel zeigt die Verwendung. `New-ELB2TargetGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine neue Zielgruppe mit den angegebenen Parametern erstellt.

```
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'
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter CreateTargetGroup AWS Tools for PowerShellCmdlet-Referenz.

`Register-ELB2Target`

Das folgende Codebeispiel zeigt die Verwendung. `Register-ELB2Target`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Instanz 'i-0672a4c4cdeae3111' bei der angegebenen Zielgruppe registriert.

```
Register-ELB2Target -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Target @{Port = 80; Id =
 'i-0672a4c4cdeae3111'}
```

- Einzelheiten zur RegisterTargets AWS Tools for PowerShellAPI finden Sie unter Cmdlet-Referenz.

`Remove-ELB2Listener`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ELB2Listener`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Listener gelöscht.

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Ausgabe:

```
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
```

Beispiel 2: In diesem Beispiel wird der angegebene Listener aus dem Load Balancer entfernt.

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618'
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DeleteListener AWS Tools for PowerShellCmdlet-Referenz.

`Remove-ELB2ListenerCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ELB2ListenerCertificate`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Zertifikat aus der angegebenen Zielgruppe entfernt.

```
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'
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter RemoveListenerCertificates AWS Tools for PowerShellCmdlet-Referenz.

`Remove-ELB2LoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ELB2LoadBalancer`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Load Balancer gelöscht.

```
Remove-ELB2LoadBalancer -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DeleteLoadBalancer AWS Tools for PowerShellCmdlet-Referenz.

`Remove-ELB2Rule`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ELB2Rule`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Regel aus dem Listener entfernt

```
Remove-ELB2Rule -RuleArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618/4b25eb10a42e33ab'
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DeleteRule AWS Tools for PowerShellCmdlet-Referenz.

`Remove-ELB2Tag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ELB2Tag`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Tag für den angegebenen Schlüssel entfernt.

```
Remove-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -TagKey
 'productVersion'
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter RemoveTags AWS Tools for PowerShellCmdlet-Referenz.

`Remove-ELB2TargetGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ELB2TargetGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Zielgruppe entfernt.

```
Remove-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/testsssss/4e0b6076bc6483a7'
```

Ausgabe:

```
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/
testsssss/4e0b6076bc6483a7".
```

```
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Einzelheiten zur API finden Sie unter DeleteTargetGroup AWS Tools for PowerShellCmdlet-Referenz.

`Set-ELB2IpAddressType`

Das folgende Codebeispiel zeigt die Verwendung. `Set-ELB2IpAddressType`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der IP-Adresstyp des Load Balancers von 'IPv4' auf '' geändert. DualStack

```
Set-ELB2IpAddressType -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -IpAddressType
 dualstack
```

Ausgabe:

```
Value
-----
dualstack
```

- Einzelheiten zur API finden Sie unter SetIpAddressType AWS Tools for PowerShellCmdlet-Referenz.

`Set-ELB2RulePriority`

Das folgende Codebeispiel zeigt die Verwendung. `Set-ELB2RulePriority`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Priorität der angegebenen Listener-Regel geändert.

```
Set-ELB2RulePriority -RulePriority -RulePriority @{Priority = 11; RuleArn =
 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/a4eb199fa5046f80/dbf4c6dcef3ec6f8'}
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter SetRulePriorities AWS Tools for PowerShellCmdlet-Referenz.

`Set-ELB2SecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Set-ELB2SecurityGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird dem angegebenen Load Balancer die Sicherheitsgruppe 'sg-07c3414abb8811cbd' hinzugefügt.

```
Set-ELB2SecurityGroup -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -SecurityGroup
 'sg-07c3414abb8811cbd'
```

Ausgabe:

```
sg-07c3414abb8811cbd
```

- Einzelheiten zur SetSecurityGroups AWS Tools for PowerShellAPI finden Sie unter Cmdlet-Referenz.

`Set-ELB2Subnet`

Das folgende Codebeispiel zeigt die Verwendung. `Set-ELB2Subnet`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Subnetze des angegebenen Load Balancers geändert.

```
Set-ELB2Subnet -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Subnet
 'subnet-7d8a0a51','subnet-c37a67a6'
```

Ausgabe:

```
LoadBalancerAddresses SubnetId        ZoneName
--------------------- --------        --------
{}                    subnet-7d8a0a51 us-east-1c
{}                    subnet-c37a67a6 us-east-1b
```

- Einzelheiten zur API finden Sie unter SetSubnetsCmdlet-Referenz.AWS Tools for PowerShell

`Unregister-ELB2Target`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-ELB2Target`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Instanz 'i-0672a4c4cdeae3111' von der angegebenen Zielgruppe abgemeldet.

```
$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'
```

- Einzelheiten zur DeregisterTargetsAPI AWS Tools for PowerShell finden Sie unter Cmdlet-Referenz.

Amazon FSx-Beispiele mit Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS Tools for PowerShell mit Amazon FSx Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zur Einrichtung und Ausführung des Codes im Kontext finden.

Themen

Aktionen

`Add-FSXResourceTag`

Das folgende Codebeispiel zeigt die Verwendung`Add-FSXResourceTag`.

Tools für PowerShell

Beispiel 1: Dieses Beispiel fügt der angegebenen Ressource Tags hinzu.

```
Add-FSXResourceTag -ResourceARN "arn:aws:fsx:eu-west-1:123456789012:file-system/
fs-01cd23bc4bdf5678a" -Tag @{Key="Users";Value="Test"} -PassThru
```

Ausgabe:

```
arn:aws:fsx:eu-west-1:123456789012:file-system/fs-01cd23bc4bdf5678a
```

- Einzelheiten zur API finden Sie unter TagResource AWS Tools for PowerShellCmdlet-Referenz.

`Get-FSXBackup`

Das folgende Codebeispiel zeigt die Verwendung. `Get-FSXBackup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Backups abgerufen, die seit gestern für die angegebene Dateisystem-ID erstellt wurden.

```
Get-FSXBackup -Filter @{Name="file-system-id";Values=$fsx.FileSystemId} | Where-
Object CreationTime -gt (Get-Date).AddDays(-1)
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeBackups AWS Tools for PowerShellCmdlet-Referenz.

`Get-FSXFileSystem`

Das folgende Codebeispiel zeigt die Verwendung. `Get-FSXFileSystem`

Tools für PowerShell

Beispiel 1: Dieses Beispiel gibt die Beschreibung der angegebenen FileSystemId zurück.

```
Get-FSXFileSystem -FileSystemId fs-01cd23bc4bdf5678a
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeFileSystemsCmdlet-Referenz.AWS Tools for PowerShell

`Get-FSXResourceTagList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-FSXResourceTagList`

Tools für PowerShell

Beispiel 1: Dieses Beispiel listet Tags für die bereitgestellte Ressource arn auf.

```
Get-FSXResourceTagList -ResourceARN $fsx.ResourceARN
```

Ausgabe:

```
Key         Value
---         -----
FSx-Service Windows
Users       Dev
```

- Einzelheiten zur API finden Sie unter ListTagsForResource AWS Tools for PowerShellCmdlet-Referenz.

`New-FSXBackup`

Das folgende Codebeispiel zeigt die Verwendung. `New-FSXBackup`

Tools für PowerShell

Beispiel 1: Dieses Beispiel erstellt eine Sicherungskopie des angegebenen Dateisystems.

```
New-FSXBackup -FileSystemId fs-0b1fac2345623456ba
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter CreateBackup AWS Tools for PowerShellCmdlet-Referenz.

`New-FSXFileSystem`

Das folgende Codebeispiel zeigt die Verwendung. `New-FSXFileSystem`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein neues 300-GB-Windows-Dateisystem erstellt, das den Zugriff vom angegebenen Subnetz aus ermöglicht und einen Durchsatz von bis zu 8 Megabyte pro Sekunde unterstützt. Das neue Dateisystem wird automatisch mit dem angegebenen Microsoft Active Directory verknüpft.

```
New-FSXFileSystem -FileSystemType WINDOWS -StorageCapacity
 300 -SubnetId subnet-1a2b3c4d5e6f -WindowsConfiguration
 @{ThroughputCapacity=8;ActiveDirectoryId='d-1a2b3c4d'}
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter CreateFileSystem AWS Tools for PowerShellCmdlet-Referenz.

`New-FSXFileSystemFromBackup`

Das folgende Codebeispiel zeigt die Verwendung. `New-FSXFileSystemFromBackup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein neues Amazon FSx-Dateisystem aus einem vorhandenen Amazon FSx for Windows File Server-Backup erstellt.

```
New-FSXFileSystemFromBackup -BackupId $backupID -Tag @{Key="tag:Name";Value="from-
manual-backup"} -SubnetId $SubnetID -SecurityGroupId $SG_ID -WindowsConfiguration
 @{ThroughputCapacity=8;ActiveDirectoryId=$DirectoryID}
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter CreateFileSystemFromBackupCmdlet-Referenz.AWS Tools for PowerShell

`Remove-FSXBackup`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-FSXBackup`

Tools für PowerShell

Beispiel 1: Dieses Beispiel entfernt die angegebene Backup-ID.

```
Remove-FSXBackup -BackupId $backupID
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DeleteBackup AWS Tools for PowerShellCmdlet-Referenz.

`Remove-FSXFileSystem`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-FSXFileSystem`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene FSX-Dateisystem-ID entfernt.

```
Remove-FSXFileSystem -FileSystemId fs-012ff34e567890120
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DeleteFileSystem AWS Tools for PowerShellCmdlet-Referenz.

`Remove-FSXResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-FSXResourceTag`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Resource-Tag für die angegebene FSX-Dateisystemressource ARN entfernt.

```
Remove-FSXResourceTag -ResourceARN $FSX.ResourceARN -TagKey Users
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter UntagResource AWS Tools for PowerShellCmdlet-Referenz.

`Update-FSXFileSystem`

Das folgende Codebeispiel zeigt die Verwendung. `Update-FSXFileSystem`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das FSX-Dateisystem über UpdateFileSystemWindowsConfiguration die automatische Aufbewahrung von Backups in Tagen aktualisiert.

```
$UpdateFSXWinConfig = [Amazon.FSx.Model.UpdateFileSystemWindowsConfiguration]::new()
 $UpdateFSXWinConfig.AutomaticBackupRetentionDays = 35
 Update-FSXFileSystem -FileSystemId $FSX.FileSystemId -WindowsConfiguration
 $UpdateFSXWinConfig
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter UpdateFileSystem AWS Tools for PowerShellCmdlet-Referenz.

AWS Glue Beispiele für die Verwendung von Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS Tools for PowerShell with Aktionen ausführen und allgemeine Szenarien implementieren AWS Glue.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

Aktionen

`New-GLUEJob`

Das folgende Codebeispiel zeigt die Verwendung`New-GLUEJob`.

Tools für PowerShell

Beispiel 1: Dieses Beispiel erstellt einen neuen Job in AWS Glue. Der Wert des Befehlsnamens ist immer**`glueetl`**. AWS Glue unterstützt das Ausführen von Jobskripten, die in Python oder Scala geschrieben wurden. In diesem Beispiel ist das Jobskript (MyTestGlueJob.py) in Python geschrieben. Python-Parameter werden in der **`$DefArgs`** Variablen angegeben und dann an den PowerShell Befehl im **`DefaultArguments`** Parameter übergeben, der eine Hashtabelle akzeptiert. Die Parameter in der **`$JobParams`** Variablen stammen aus der CreateJob API, die im Thema Jobs (https://docs.aws.amazon.com/glue/latest/dg/ aws-glue-api-jobs -job.html) der AWS Glue-API-Referenz dokumentiert ist.

```
$Command = New-Object Amazon.Glue.Model.JobCommand
$Command.Name = 'glueetl'
$Command.ScriptLocation = 's3://aws-glue-scripts-000000000000-us-west-2/admin/
MyTestGlueJob.py'
```

```
$Command

$Source = "source_test_table"
$Target = "target_test_table"
$Connections = $Source, $Target

$DefArgs = @{
     '--TempDir' = 's3://aws-glue-temporary-000000000000-us-west-2/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
```

- Einzelheiten zur API finden Sie unter CreateJob AWS Tools for PowerShellCmdlet-Referenz.

AWS Health Beispiele für die Verwendung von Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS Tools for PowerShell with Aktionen ausführen und allgemeine Szenarien implementieren AWS Health.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

Aktionen

`Get-HLTHEvent`

Das folgende Codebeispiel zeigt die Verwendung`Get-HLTHEvent`.

Tools für PowerShell

Beispiel 1: Dieser Befehl gibt Ereignisse aus dem AWS Personal Health Dashboard zurück. Der Benutzer fügt den Parameter -Region hinzu, um Ereignisse anzuzeigen, die für den Service in der Region USA Ost (Nord-Virginia) verfügbar sind. Der Parameter -Filter_Region filtert jedoch nach Ereignissen, die in den Regionen EU (London) und USA West (Oregon) (eu-west-2 und us-west-2) protokolliert werden. Der StartTime Parameter -Filter_ filtert nach einem bestimmten Zeitbereich, zu dem Ereignisse beginnen können, während der Parameter -Filter_ nach einem bestimmten Zeitbereich filtertEndTime , zu dem Ereignisse enden können. Das Ergebnis ist ein geplantes Wartungsereignis für RDS, das innerhalb des angegebenen -Filter_-Bereichs beginnt und innerhalb des geplanten StartTime -Filter_-Bereichs endet. EndTime

```
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"}
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeEventsCmdlet-Referenz.AWS Tools for PowerShell

IAM-Beispiele mit Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS Tools for PowerShell mit IAM Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`Add-IAMClientIDToOpenIDConnectProvider`

Das folgende Codebeispiel zeigt die VerwendungAdd-IAMClientIDToOpenIDConnectProvider.

Tools für PowerShell

Beispiel 1: Dieser Befehl fügt die Client-ID (oder Audience) **my-application-ID** dem vorhandenen OIDC-Anbieter namens hinzu. **server.example.com**

```
Add-IAMClientIDToOpenIDConnectProvider -ClientID "my-application-ID"
 -OpenIDConnectProviderARN "arn:aws:iam::123456789012:oidc-provider/
server.example.com"
```

- Einzelheiten zur API finden Sie unter AddClientIdToOpenIdConnectProvider AWS Tools for PowerShellCmdlet-Referenz.

`Add-IAMRoleTag`

Das folgende Codebeispiel zeigt die Verwendung. `Add-IAMRoleTag`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der Rolle im Identity Management Service ein Tag hinzugefügt

```
Add-IAMRoleTag -RoleName AdminRoleacess -Tag @{ Key = 'abac'; Value = 'testing'}
```

- Einzelheiten zur API finden Sie unter TagRole AWS Tools for PowerShellCmdlet-Referenz.

`Add-IAMRoleToInstanceProfile`

Das folgende Codebeispiel zeigt die Verwendung. `Add-IAMRoleToInstanceProfile`

Tools für PowerShell

Beispiel 1: Dieser Befehl fügt die angegebene Rolle einem vorhandenen Instanzprofil mit dem Namen hinzu**`webserver`**. **`S3Access`** Verwenden Sie den **`New-IAMInstanceProfile`** Befehl, um das Instanzprofil zu erstellen. Nachdem Sie das Instanzprofil erstellt und es mithilfe dieses Befehls einer Rolle zugeordnet haben, können Sie es an eine EC2-Instance anhängen. Verwenden Sie dazu das **`New-EC2Instance`** Cmdlet mit dem **`InstanceProfile-Name`** Parameter **`InstanceProfile_Arn`** oder, um die neue Instance zu starten.

```
Add-IAMRoleToInstanceProfile -RoleName "S3Access" -InstanceProfileName "webserver"
```

- Einzelheiten zur API finden Sie unter AddRoleToInstanceProfile AWS Tools for PowerShellCmdlet-Referenz.

`Add-IAMUserTag`

Das folgende Codebeispiel zeigt die Verwendung. `Add-IAMUserTag`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird dem Benutzer im Identity Management Service ein Tag hinzugefügt

```
Add-IAMUserTag -UserName joe -Tag @{ Key = 'abac'; Value = 'testing'}
```

- Einzelheiten zur API finden Sie unter TagUser AWS Tools for PowerShellCmdlet-Referenz.

`Add-IAMUserToGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Add-IAMUserToGroup`

Tools für PowerShell

Beispiel 1: Mit diesem Befehl wird der angegebene Benutzer **`Bob`** zur Gruppe mit dem Namen hinzugefügt**`Admins`**.

```
Add-IAMUserToGroup -UserName "Bob" -GroupName "Admins"
```

- Einzelheiten zur API finden Sie unter AddUserToGroup AWS Tools for PowerShellCmdlet-Referenz.

`Disable-IAMMFADevice`

Das folgende Codebeispiel zeigt die Verwendung. `Disable-IAMMFADevice`

Tools für PowerShell

Beispiel 1: Dieser Befehl deaktiviert das Hardware-MFA-Gerät, das dem Benutzer zugeordnet ist**`Bob`**, der die Seriennummer besitzt. **`123456789012`**

```
Disable-IAMMFADevice -UserName "Bob" -SerialNumber "123456789012"
```

Beispiel 2: Dieser Befehl deaktiviert das virtuelle MFA-Gerät, das dem Benutzer zugeordnet ist**`David`**, der über den ARN verfügt. **`arn:aws:iam::210987654321:mfa/David`** Beachten

Sie, dass das virtuelle MFA-Gerät nicht aus dem Konto gelöscht wird. Das virtuelle Gerät ist immer noch vorhanden und erscheint in der Ausgabe des **`Get-IAMVirtualMFADevice`** Befehls. Bevor Sie ein neues virtuelles MFA-Gerät für denselben Benutzer erstellen können, müssen Sie das alte mit dem **`Remove-IAMVirtualMFADevice`** Befehl löschen.

```
Disable-IAMMFADevice -UserName "David" -SerialNumber "arn:aws:iam::210987654321:mfa/
David"
```

- Einzelheiten zur API finden Sie unter DeactivateMfaDevice AWS Tools for PowerShellCmdlet-Referenz.

`Edit-IAMPassword`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-IAMPassword`

Tools für PowerShell

Beispiel 1: Dieser Befehl ändert das Passwort für den Benutzer, der den Befehl ausführt. Dieser Befehl kann nur von IAM-Benutzern aufgerufen werden. Wenn dieser Befehl aufgerufen wird, während Sie mit AWS Kontoanmeldeinformationen (Root) angemeldet sind, gibt der Befehl einen Fehler zurück. **`InvalidUserType`**

```
Edit-IAMPassword -OldPassword "MyOldP@ssw0rd" -NewPassword "MyNewP@ssw0rd"
```

- Einzelheiten zur API finden Sie unter ChangePasswordCmdlet-Referenz.AWS Tools for PowerShell

`Enable-IAMMFADevice`

Das folgende Codebeispiel zeigt die Verwendung. `Enable-IAMMFADevice`

Tools für PowerShell

Beispiel 1: Dieser Befehl aktiviert das Hardware-MFA-Gerät mit der Seriennummer **`987654321098`** und ordnet das Gerät dem Benutzer **`Bob`** zu. Er enthält nacheinander die ersten beiden Codes des Geräts.

```
Enable-IAMMFADevice -UserName "Bob" -SerialNumber "987654321098" -
AuthenticationCode1 "12345678" -AuthenticationCode2 "87654321"
```

Beispiel 2: In diesem Beispiel wird ein virtuelles MFA-Gerät erstellt und aktiviert. Der erste Befehl erstellt das virtuelle Gerät und gibt die Objektdarstellung des Geräts in der Variablen **$MFADevice** zurück. Sie können die **QRCodePng** Eigenschaften **.Base32StringSeed** oder verwenden, um die Softwareanwendung des Benutzers zu konfigurieren. Mit dem letzten Befehl wird das Gerät dem Benutzer zugewiesen **David** und das Gerät anhand seiner Seriennummer identifiziert. Der Befehl synchronisiert das Gerät auch mit, AWS indem er die ersten beiden Codes nacheinander vom virtuellen MFA-Gerät einfügt.

```
$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"
```

- Einzelheiten zur API finden Sie unter EnableMfaDevice AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMAccessKey`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMAccessKey`

Tools für PowerShell

Beispiel 1: Dieser Befehl listet die Zugriffsschlüssel für den IAM-Benutzer mit dem Namen **Bob** auf. Beachten Sie, dass Sie die geheimen Zugriffsschlüssel für IAM-Benutzer nicht auflisten können. Wenn die geheimen Zugriffsschlüssel verloren gehen, müssen Sie mit dem **New-IAMAccessKey** Cmdlet neue Zugriffsschlüssel erstellen.

```
Get-IAMAccessKey -UserName "Bob"
```

Ausgabe:

```
AccessKeyId              CreateDate                Status              UserName
-----------              ----------                ------              --------
AKIAIOSFODNN7EXAMPLE     12/3/2014 10:53:41 AM     Active              Bob
AKIAI44QH8DHBEXAMPLE     6/6/2013 8:42:26 PM       Inactive            Bob
```

- Einzelheiten zur API finden Sie unter ListAccessKeys AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMAccessKeyLastUsed`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMAccessKeyLastUsed`

Tools für PowerShell

Beispiel 1: Gibt den Namen des Besitzers und Informationen zur letzten Verwendung des angegebenen Zugriffsschlüssels zurück.

```
Get-IAMAccessKeyLastUsed -AccessKeyId ABCDEXAMPLE
```

- Einzelheiten zur API finden Sie unter GetAccessKeyLastUsedCmdlet-Referenz.AWS Tools for PowerShell

`Get-IAMAccountAlias`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMAccountAlias`

Tools für PowerShell

Beispiel 1: Dieser Befehl gibt den Kontoalias für den zurück AWS-Konto.

```
Get-IAMAccountAlias
```

Ausgabe:

```
ExampleCo
```

- Einzelheiten zur API finden Sie unter ListAccountAliases AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMAccountAuthorizationDetail`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMAccountAuthorizationDetail`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Autorisierungsdetails zu den Identitäten im AWS Konto abgerufen und die Elementliste des zurückgegebenen Objekts angezeigt, einschließlich Benutzer,

Gruppen und Rollen. In der **`UserDetailList`** Eigenschaft werden beispielsweise Details zu den Benutzern angezeigt. Ähnliche Informationen sind in den **`GroupDetailList`** Eigenschaften **`RoleDetailList`** und verfügbar.

```
$Details=Get-IAMAccountAuthorizationDetail
$Details
```

Ausgabe:

```
GroupDetailList : {Administrators, Developers, Testers, Backup}
IsTruncated     : False
Marker          :
RoleDetailList  : {TestRole1, AdminRole, TesterRole, clirole...}
UserDetailList  : {Administrator, Bob, BackupToS3, }
```

```
$Details.UserDetailList
```

Ausgabe:

```
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}
```

- Einzelheiten zur API finden Sie unter GetAccountAuthorizationDetails AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMAccountPasswordPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMAccountPasswordPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Details zur Passwortrichtlinie für das aktuelle Konto zurückgegeben. Wenn keine Passwortrichtlinie für das Konto definiert ist, gibt der Befehl einen **`NoSuchEntity`** Fehler zurück.

```
Get-IAMAccountPasswordPolicy
```

Ausgabe:

```
AllowUsersToChangePassword : True
ExpirePasswords            : True
HardExpiry                 : False
MaxPasswordAge             : 90
MinimumPasswordLength      : 8
PasswordReusePrevention    : 20
RequireLowercaseCharacters : True
RequireNumbers             : True
RequireSymbols             : False
RequireUppercaseCharacters : True
```

- Einzelheiten zur API finden Sie unter GetAccountPasswordPolicy AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMAccountSummary`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMAccountSummary`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Informationen zur aktuellen Nutzung der IAM-Entität und zu den aktuellen IAM-Entitätskontingenten in der zurückgegeben. AWS-Konto

```
Get-IAMAccountSummary
```

Ausgabe:

```
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
...
```

- Einzelheiten zur API finden Sie unter GetAccountSummary AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMAttachedGroupPolicyList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMAttachedGroupPolicyList`

Tools für PowerShell

Beispiel 1: Dieser Befehl gibt die Namen und ARNs der verwalteten Richtlinien zurück, die der **`Admins`** im Konto genannten IAM-Gruppe zugeordnet sind. AWS Verwenden Sie den Befehl, um die Liste der in die Gruppe eingebetteten Inline-Richtlinien anzuzeigen. **`Get-IAMGroupPolicyList`**

```
Get-IAMAttachedGroupPolicyList -GroupName "Admins"
```

Ausgabe:

```
PolicyArn                                                 PolicyName
---------                                                 ----------
arn:aws:iam::aws:policy/SecurityAudit                     SecurityAudit
arn:aws:iam::aws:policy/AdministratorAccess               AdministratorAccess
```

- Einzelheiten zur API finden Sie unter ListAttachedGroupPolicies AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMAttachedRolePolicyList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMAttachedRolePolicyList`

Tools für PowerShell

Beispiel 1: Dieser Befehl gibt die Namen und ARNs der verwalteten Richtlinien zurück, die der **`SecurityAuditRole`** im Konto genannten IAM-Rolle zugeordnet sind. AWS Verwenden Sie den Befehl, um die Liste der Inline-Richtlinien anzuzeigen, die in die Rolle eingebettet sind. **`Get-IAMRolePolicyList`**

```
Get-IAMAttachedRolePolicyList -RoleName "SecurityAuditRole"
```

Ausgabe:

```
PolicyArn                                                 PolicyName
---------                                                 ----------
```

```
arn:aws:iam::aws:policy/SecurityAudit                    SecurityAudit
```

- Einzelheiten zur API finden Sie unter ListAttachedRolePolicies AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMAttachedUserPolicyList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMAttachedUserPolicyList`

Tools für PowerShell

Beispiel 1: Dieser Befehl gibt die Namen und ARNs der verwalteten Richtlinien für den IAM-Benutzer zurück, der **Bob** AWS im Konto angegeben ist. Verwenden Sie den Befehl, um die Liste der Inline-Richtlinien anzuzeigen, die in den IAM-Benutzer eingebettet sind. **`Get-IAMUserPolicyList`**

```
Get-IAMAttachedUserPolicyList -UserName "Bob"
```

Ausgabe:

```
PolicyArn                                                PolicyName
---------                                                ----------
arn:aws:iam::aws:policy/TesterPolicy                     TesterPolicy
```

- Einzelheiten zur API finden Sie unter ListAttachedUserPolicies AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMContextKeysForCustomPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMContextKeysForCustomPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden alle Kontextschlüssel abgerufen, die in der bereitgestellten Richtlinien-JSON vorhanden sind. Um mehrere Richtlinien bereitzustellen, können Sie sie als kommagetrennte Werteliste angeben.

```
$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
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz. GetContextKeysForCustomPolicyAWS Tools for PowerShell

`Get-IAMContextKeysForPrincipalPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMContextKeysForPrincipalPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden alle Kontextschlüssel abgerufen, die in der bereitgestellten Policy-JSON-Datei und in den Richtlinien enthalten sind, die der IAM-Entität zugeordnet sind (Benutzer/Rolle usw.). Für — PolicyInputList Sie können eine Liste mit mehreren Werten als durch Kommas getrennte Werte angeben.

```
$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
```

- Einzelheiten zur API finden Sie unter GetContextKeysForPrincipalPolicy AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMCredentialReport`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMCredentialReport`

Beispiel 1: In diesem Beispiel wird der zurückgegebene Bericht geöffnet und als Array von Textzeilen an die Pipeline ausgegeben. Die erste Zeile ist die Kopfzeile mit durch Kommas getrennten Spaltennamen. Jede nachfolgende Zeile ist die Detailzeile für einen Benutzer, wobei jedes Feld durch Kommas getrennt ist. Bevor Sie den Bericht anzeigen können, müssen Sie ihn mit dem **`Request-IAMCredentialReport`** Cmdlet generieren. Um den Bericht als einzelne Zeichenfolge abzurufen, verwenden Sie **`-Raw`** anstelle von. **`-AsTextArray`** Der Alias **`-SplitLines`** wird auch für den **`-AsTextArray`** Switch akzeptiert. Die vollständige Liste der Spalten in der Ausgabe finden Sie in der Service-API-Referenz. Beachten Sie, dass Sie, wenn Sie **`-AsTextArray`** oder nicht verwenden**`-SplitLines`**, den Text mithilfe der **`StreamReader`** .NET-Klasse aus der **`.Content`** Eigenschaft extrahieren müssen.

```
Request-IAMCredentialReport
```

Ausgabe:

```
Description                                                    State
-----------                                                    -----
No report exists. Starting a new report generation task        STARTED
```

```
Get-IAMCredentialReport -AsTextArray
```

Ausgabe:

```
 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
```

- Einzelheiten zur API finden Sie unter GetCredentialReport AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMEntitiesForPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMEntitiesForPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste von IAM-Gruppen, -Rollen und Benutzern zurückgegeben, denen die Richtlinie **`arn:aws:iam::123456789012:policy/TestPolicy`** zugewiesen wurde.

```
Get-IAMEntitiesForPolicy -PolicyArn "arn:aws:iam::123456789012:policy/TestPolicy"
```

Ausgabe:

```
IsTruncated  : False
Marker       :
PolicyGroups : {}
PolicyRoles  : {testRole}
PolicyUsers  : {Bob, Theresa}
```

- Einzelheiten zur API finden Sie unter ListEntitiesForPolicy AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Details zur IAM-Gruppe zurückgegeben**`Testers`**, einschließlich einer Sammlung aller IAM-Benutzer, die zu der Gruppe gehören.

```
$results = Get-IAMGroup -GroupName "Testers"
$results
```

Ausgabe:

```
Group                                    IsTruncated           Marker
  Users
-----                                    -----------           ------
  -----
```

```
Amazon.IdentityManagement.Model.Group     False
  {Theresa, David}
```

```
$results.Group
```

Ausgabe:

```
Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : 3RHNZZGQJ7QHMAEXAMPLE1
GroupName  : Testers
Path       : /
```

```
$results.Users
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter GetGroup AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMGroupForUser`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMGroupForUser`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Liste der IAM-Gruppen zurückgegeben, zu denen der IAM-Benutzer **`David`** gehört.

```
Get-IAMGroupForUser -UserName David
```

Ausgabe:

```
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       : /
```

- Einzelheiten zur API finden Sie unter ListGroupsForUser AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMGroupList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMGroupList`

Tools für PowerShell

Beispiel 1: Dieses Beispiel gibt eine Sammlung aller in der aktuellen AWS-Konto Version definierten IAM-Gruppen zurück.

```
Get-IAMGroupList
```

Ausgabe:

```
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       : /
```

- Einzelheiten zur API finden Sie unter ListGroups AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMGroupPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMGroupPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Details zur eingebetteten Inline-Richtlinie zurückgegeben, die **`PowerUserAccess-Testers`** nach der Gruppe benannt ist**`Testers`**. Die **`PolicyDocument`** Eigenschaft ist URL-codiert. Sie wird in diesem Beispiel mit der **`UrlDecode`** .NET-Methode dekodiert.

```
$results = Get-IAMGroupPolicy -GroupName Testers -PolicyName PowerUserAccess-Testers
$results
```

Ausgabe:

```
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": "*"
    }
  ]
}
```

- Einzelheiten zur API finden Sie unter GetGroupPolicy AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMGroupPolicyList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMGroupPolicyList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste der Inline-Richtlinien zurückgegeben, die in die Gruppe eingebettet sind**`Testers`**. Verwenden Sie den Befehl, um die verwalteten Richtlinien abzurufen, die der Gruppe zugeordnet sind**`Get-IAMAttachedGroupPolicyList`**.

```
Get-IAMGroupPolicyList -GroupName Testers
```

Ausgabe:

```
Deny-Assume-S3-Role-In-Production
PowerUserAccess-Testers
```

- Einzelheiten zur API finden Sie unter ListGroupPolicies AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMInstanceProfile`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMInstanceProfile`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Details des genannten Instanzprofils zurückgegeben**`ec2instancerole`**, das im aktuellen AWS Konto definiert ist.

```
Get-IAMInstanceProfile -InstanceProfileName ec2instancerole
```

Ausgabe:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- Einzelheiten zur API finden Sie unter GetInstanceProfile AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMInstanceProfileForRole`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMInstanceProfileForRole`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Details des mit der Rolle verknüpften Instanzprofils zurückgegeben**`ec2instancerole`**.

```
Get-IAMInstanceProfileForRole -RoleName ec2instancerole
```

Ausgabe:

```
      Arn                 : arn:aws:iam::123456789012:instance-profile/
ec2instancerole
      CreateDate          : 2/17/2015 2:49:04 PM
      InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
      InstanceProfileName : ec2instancerole
      Path                : /
      Roles               : {ec2instancerole}
```

- Einzelheiten zur API finden Sie unter ListInstanceProfilesForRole AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMInstanceProfileList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMInstanceProfileList`

Tools für PowerShell

Beispiel 1: Dieses Beispiel gibt eine Sammlung der Instanzprofile zurück, die in der aktuellen Version definiert sind AWS-Konto.

```
Get-IAMInstanceProfileList
```

Ausgabe:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- Einzelheiten zur API finden Sie unter ListInstanceProfiles AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMLoginProfile`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMLoginProfile`

Tools für PowerShell

Beispiel 1: Dieses Beispiel gibt das Erstellungsdatum des Kennworts zurück und gibt an, ob für den IAM-Benutzer **`David`** ein Zurücksetzen des Kennworts erforderlich ist.

```
Get-IAMLoginProfile -UserName David
```

Ausgabe:

```
CreateDate                PasswordResetRequired                UserName
----------                ---------------------                --------
12/10/2014 3:39:44 PM     False                                David
```

- Einzelheiten zur API finden Sie unter GetLoginProfile AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMMFADevice`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMMFADevice`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Details über das MFA-Gerät zurückgegeben, das dem IAM-Benutzer zugewiesen wurde. **`David`** In diesem Beispiel können Sie erkennen, dass es sich um ein virtuelles Gerät **`SerialNumber`** handelt, da es sich um einen ARN und nicht um die tatsächliche Seriennummer eines physischen Geräts handelt.

```
Get-IAMMFADevice -UserName David
```

Ausgabe:

```
EnableDate                SerialNumber                          UserName
----------                ------------                          --------
4/8/2015 9:41:10 AM       arn:aws:iam::123456789012:mfa/David   David
```

- Einzelheiten zur API finden Sie unter ListMfaDevices AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMOpenIDConnectProvider`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMOpenIDConnectProvider`

Tools für PowerShell

Beispiel 1: Dieses Beispiel gibt Details über den OpenID Connect-Anbieter zurück, dessen ARN lautet**`arn:aws:iam::123456789012:oidc-provider/accounts.google.com`**. Die **`ClientIDList`** Eigenschaft ist eine Sammlung, die alle für diesen Anbieter definierten Client-IDs enthält.

```
Get-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/oidc.example.com
```

Ausgabe:

```
ClientIDList         CreateDate               ThumbprintList
        Url
------------         ----------               --------------
        ---
{MyOIDCApp}          2/3/2015 3:00:30 PM
 {12345abcdefghijk67890lmnopqrst98765uvwxy}   oidc.example.com
```

- Einzelheiten zur API finden Sie unter GetOpenIdConnectProvider AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMOpenIDConnectProviderList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMOpenIDConnectProviderList`

Tools für PowerShell

Beispiel 1: Dieses Beispiel gibt eine Liste von ARNS aller OpenID Connect-Anbieter zurück, die in der aktuellen Version definiert sind. AWS-Konto

```
Get-IAMOpenIDConnectProviderList
```

Ausgabe:

```
Arn
---
arn:aws:iam::123456789012:oidc-provider/server.example.com
arn:aws:iam::123456789012:oidc-provider/another.provider.com
```

- Einzelheiten zur API finden Sie unter ListOpenIdConnectProviders AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Details zu der verwalteten Richtlinie zurückgegeben, deren ARN lautet**`arn:aws:iam::123456789012:policy/MySamplePolicy`**.

```
Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter GetPolicy AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMPolicyList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMPolicyList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Sammlung der ersten drei verwalteten Richtlinien zurückgegeben, die im aktuellen AWS Konto verfügbar sind. Da nicht angegeben **`-scope`** ist, werden standardmäßig sowohl AWS verwaltete als auch vom **`all`** Kunden verwaltete Richtlinien verwendet und diese sind auch enthalten.

```
Get-IAMPolicyList -MaxItem 3
```

Ausgabe:

```
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
```

Beispiel 2: In diesem Beispiel wird eine Sammlung der ersten beiden vom Kunden verwalteten Policen zurückgegeben, die im AWS Girokonto verfügbar sind. Es wird verwendet**`-Scope local`**, um die Ausgabe nur auf vom Kunden verwaltete Policen zu beschränken.

```
Get-IAMPolicyList -Scope local -MaxItem 2
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter ListPolicies AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMPolicyVersion`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMPolicyVersion`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Richtliniendokument für die **v2** Version der Richtlinie zurückgegeben, deren ARN lautet**arn:aws:iam::123456789012:policy/MyManagedPolicy**. Das Richtliniendokument in der **Document** Eigenschaft ist URL-kodiert und wird in diesem Beispiel mit der **UrlDecode** .NET-Methode dekodiert.

```
$results = Get-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/
MyManagedPolicy -VersionId v2
$results
```

Ausgabe:

```
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": "*"
    }
}
```

- Einzelheiten zur API finden Sie unter GetPolicyVersion AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMPolicyVersionList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMPolicyVersionList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Liste der verfügbaren Versionen der Richtlinie zurückgegeben, deren ARN lautet**`arn:aws:iam::123456789012:policy/MyManagedPolicy`**. Um das Richtliniendokument für eine bestimmte Version abzurufen, verwenden Sie den **`Get-IAMPolicyVersion`** Befehl und geben Sie die gewünschte Version an. **`VersionId`**

```
Get-IAMPolicyVersionList -PolicyArn arn:aws:iam::123456789012:policy/MyManagedPolicy
```

Ausgabe:

```
CreateDate                 Document                IsDefaultVersion
    VersionId
----------                 --------                ----------------
    ---------
2/12/2015 9:39:53 AM                               True
    v2
2/12/2015 9:39:09 AM                               False
    v1
```

- Einzelheiten zur API finden Sie unter ListPolicyVersions AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMRole`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMRole`

Tools für PowerShell

Beispiel 1: Dieses Beispiel gibt die Details von zurück**`lamda_exec_role`**. Es enthält das Dokument mit der Vertrauensrichtlinie, in dem angegeben ist, wer diese Rolle übernehmen kann. Das Richtliniendokument ist URL-kodiert und kann mit der **`UrlDecode`** .NET-Methode dekodiert werden. In diesem Beispiel wurden bei der ursprünglichen Richtlinie alle Leerzeichen entfernt, bevor sie in die Richtlinie hochgeladen wurde. Um die Dokumente mit den Berechtigungsrichtlinien einzusehen, in denen festgelegt ist, was jemand, der die Rolle übernimmt, tun kann, verwenden Sie die Option **`Get-IAMRolePolicy`** für Inline-Richtlinien und **`Get-IAMPolicyVersion`** für angefügte verwaltete Richtlinien.

```
$results = Get-IamRole -RoleName lambda_exec_role
$results | Format-List
```

Ausgabe:

```
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
```

Ausgabe:

```
{"Version":"2012-10-17","Statement":[{"Sid":"","Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}
```

- Einzelheiten zur API finden Sie unter GetRole AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMRoleList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMRoleList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste aller IAM-Rollen in der abgerufen. AWS-Konto

```
Get-IAMRoleList
```

Beispiel 2: In diesem Beispielcodeausschnitt wird eine Liste von IAM-Rollen im AWS Konto abgerufen und jeweils drei Rollen angezeigt. Anschließend wird darauf gewartet, dass Sie zwischen den einzelnen Gruppen die Eingabetaste drücken. Es übergibt den **`Marker`** Wert des vorherigen Aufrufs, um anzugeben, wo die nächste Gruppe beginnen soll.

```
$nextMarker = $null
Do
{
    $results = Get-IAMRoleList -MaxItem 3 -Marker $nextMarker
    $nextMarker = $AWSHistory.LastServiceResponse.Marker
    $results
    Read-Host
} while ($nextMarker -ne $null)
```

- Einzelheiten zur API finden Sie unter ListRoles AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMRolePolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Dokument mit der Berechtigungsrichtlinie für die angegebene Richtlinie zurückgegeben**`oneClick_lambda_exec_role_policy`**, die in die IAM-Rolle **`lamda_exec_role`** eingebettet ist. Das resultierende Richtliniendokument ist URL-kodiert. In diesem Beispiel wird es mit der **`UrlDecode`** .NET-Methode dekodiert.

```
$results = Get-IAMRolePolicy -RoleName lambda_exec_role -PolicyName
 oneClick_lambda_exec_role_policy
$results
```

Ausgabe:

```
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)
```

Ausgabe:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:*"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::*"
      ]
    }
  ]
}
```

- Einzelheiten zur API finden Sie unter GetRolePolicy AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMRolePolicyList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMRolePolicyList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Liste der Namen von Inline-Richtlinien zurückgegeben, die in die IAM-Rolle **`lamda_exec_role`** eingebettet sind. Verwenden Sie den Befehl**`Get-IAMRolePolicy`**, um die Details einer Inline-Richtlinie anzuzeigen.

```
Get-IAMRolePolicyList -RoleName lambda_exec_role
```

Ausgabe:

```
oneClick_lambda_exec_role_policy
```

- Einzelheiten zur API finden Sie unter ListRolePolicies AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMRoleTagList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMRoleTagList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das der Rolle zugeordnete Tag abgerufen..

```
Get-IAMRoleTagList -RoleName MyRoleName
```

- Einzelheiten zur API finden Sie unter ListRoleTags AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMSAMLProvider`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMSAMLProvider`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Details zum SAML 2.0-Anbieter abgerufen, dessen ARM arn:aws:iam: :123456789012:SAML-Provider/samladfs ist. Die Antwort enthält das Metadatendokument, das Sie vom Identitätsanbieter zur Erstellung der SAML-Provider-Entität erhalten haben, sowie die Erstellungs- und Ablaufdaten. AWS

```
Get-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS
```

Ausgabe:

```
CreateDate                 SAMLMetadataDocument
     ValidUntil
----------                 --------------------
     ----------
12/23/2014 12:16:55 PM    <EntityDescriptor ID="_12345678-1234-5678-9012-example1...
    12/23/2114 12:16:54 PM
```

- Einzelheiten zur API finden Sie unter GetSamlProvider AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMSAMLProviderList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMSAMLProviderList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Liste der SAML 2.0-Anbieter abgerufen, die in der aktuellen Version erstellt wurden. AWS-Konto Es gibt den ARN, das Erstellungs- und das Ablaufdatum für jeden SAML-Anbieter zurück.

```
Get-IAMSAMLProviderList
```

Ausgabe:

```
Arn                                                CreateDate
 ValidUntil
---                                                ----------
 ----------
```

```
arn:aws:iam::123456789012:saml-provider/SAMLADFS    12/23/2014 12:16:55 PM
 12/23/2114 12:16:54 PM
```

- Einzelheiten zur API finden Sie unter ListSamlProviders in der Cmdlet-Referenz.AWS Tools for PowerShell

`Get-IAMServerCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMServerCertificate`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Details über das angegebene Serverzertifikat abgerufen. **`MyServerCertificate`** Sie finden die Zertifikatsdetails in den **`ServerCertificateMetadata`** Eigenschaften **`CertificateBody`** und.

```
$result = Get-IAMServerCertificate -ServerCertificateName MyServerCertificate
$result | format-list
```

Ausgabe:

```
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
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter GetServerCertificate AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMServerCertificateList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMServerCertificateList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Liste der Serverzertifikate abgerufen, die auf den aktuellen AWS-Konto Server hochgeladen wurden.

```
Get-IAMServerCertificateList
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter ListServerCertificates AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMServiceLastAccessedDetail`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMServiceLastAccessedDetail`

Tools für PowerShell

Beispiel 1: Dieses Beispiel enthält Details zum Dienst, auf den die IAM-Entität (Benutzer, Gruppe, Rolle oder Richtlinie) zuletzt zugegriffen hat, die dem Anforderungsaufruf zugeordnet ist.

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

Ausgabe:

```
f0b7a819-eab0-929b-dc26-ca598911cb9f
```

```
Get-IAMServiceLastAccessedDetail -JobId f0b7a819-eab0-929b-dc26-ca598911cb9f
```

- Einzelheiten zur API finden Sie unter GetServiceLastAccessedDetails AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMServiceLastAccessedDetailWithEntity`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMServiceLastAccessedDetailWithEntity`

Tools für PowerShell

Beispiel 1: Dieses Beispiel liefert den Zeitstempel, auf den zuletzt zugegriffen wurde, für den Dienst in der Anfrage der jeweiligen IAM-Entität.

```
$results = Get-IAMServiceLastAccessedDetailWithEntity -JobId f0b7a819-eab0-929b-
dc26-ca598911cb9f -ServiceNamespace ec2
$results
```

Ausgabe:

```
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
```

Ausgabe:

```
EntityInfo                                  LastAuthenticated
----------                                  -----------------
Amazon.IdentityManagement.Model.EntityInfo 11/16/19 3:47:00 PM
```

```
$results.EntityInfo
```

Ausgabe:

```
Arn  : arn:aws:iam::123456789012:user/TestUser
Id   : AIDA4NBK5CXF5TZHU1234
Name : TestUser
Path : /
Type : USER
```

- Einzelheiten zur API finden Sie unter GetServiceLastAccessedDetailsWithEntities AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMSigningCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMSigningCertificate`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Details über das Signaturzertifikat abgerufen, das dem genannten **Bob** Benutzer zugeordnet ist.

```
Get-IAMSigningCertificate -UserName Bob
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter ListSigningCertificates AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMUser`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMUser`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Details über den genannten **David** Benutzer abgerufen.

```
Get-IAMUser -UserName David
```

Ausgabe:

```
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
```

Beispiel 2: In diesem Beispiel werden Details über den aktuell angemeldeten IAM-Benutzer abgerufen.

```
Get-IAMUser
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz. GetUserAWS Tools for PowerShell

`Get-IAMUserList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMUserList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Sammlung von Benutzern in der aktuellen AWS-Konto Version abgerufen.

```
Get-IAMUserList
```

Ausgabe:

```
    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
```

- Einzelheiten zur API finden Sie unter ListUsers AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMUserPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Details der genannten Inline-Richtlinie abgerufen**`Davids_IAM_Admin_Policy`**, die in den IAM-Benutzer mit dem Namen eingebettet ist. **`David`** Das Richtliniendokument ist URL-codiert.

```
$results = Get-IAMUserPolicy -PolicyName Davids_IAM_Admin_Policy -UserName David
$results
```

Ausgabe:

```
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": [
        "*"
      ]
    }
  ]
}
```

- Einzelheiten zur API finden Sie unter GetUserPolicy AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMUserPolicyList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMUserPolicyList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Liste der Namen der Inline-Richtlinien abgerufen, die in den IAM-Benutzer namens eingebettet sind. **`David`**

```
Get-IAMUserPolicyList -UserName David
```

Ausgabe:

```
Davids_IAM_Admin_Policy
```

- Einzelheiten zur API finden Sie unter ListUserPolicies AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMUserTagList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMUserTagList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das dem Benutzer zugeordnete Tag abgerufen.

```
Get-IAMUserTagList -UserName joe
```

- Einzelheiten zur API finden Sie unter ListUserTags AWS Tools for PowerShellCmdlet-Referenz.

`Get-IAMVirtualMFADevice`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMVirtualMFADevice`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Sammlung der virtuellen MFA-Geräte abgerufen, die Benutzern im AWS Konto zugewiesen sind. Bei jeder **`User`** Eigenschaft handelt es sich um ein Objekt mit Angaben zum IAM-Benutzer, dem das Gerät zugewiesen ist.

```
Get-IAMVirtualMFADevice -AssignmentStatus Assigned
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter ListVirtualMfaDevices AWS Tools for PowerShellCmdlet-Referenz.

`New-IAMAccessKey`

Das folgende Codebeispiel zeigt die Verwendung. `New-IAMAccessKey`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein neues Paar aus Zugriffsschlüssel und geheimem Zugriffsschlüssel erstellt und dem Benutzer **`David`** zugewiesen. Stellen Sie sicher, dass Sie die **`SecretAccessKey`** Werte **`AccessKeyId`** und in einer Datei speichern, da Sie die nur zu diesem Zeitpunkt abrufen können. **`SecretAccessKey`** Sie können es später nicht abrufen. Wenn Sie den geheimen Schlüssel verlieren, müssen Sie ein neues Zugriffsschlüsselpaar erstellen.

```
New-IAMAccessKey -UserName David
```

Ausgabe:

```
AccessKeyId     : AKIAIOSFODNN7EXAMPLE
CreateDate      : 4/13/2015 1:00:42 PM
SecretAccessKey : wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Status          : Active
UserName        : David
```

- Einzelheiten zur API finden Sie unter CreateAccessKey AWS Tools for PowerShellCmdlet-Referenz.

`New-IAMAccountAlias`

Das folgende Codebeispiel zeigt die Verwendung. `New-IAMAccountAlias`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der Kontoalias für Ihr AWS Konto in geändert**`mycompanyaws`**. Die Adresse der Benutzeranmeldeseite wird in https://mycompanyaws.signin.aws.amazon.com/console geändert. Die ursprüngliche URL, die Ihre Konto-ID-Nummer anstelle des Alias verwendet (https://<accountidnumber>.signin.aws.amazon.com/console), funktioniert weiterhin. Alle zuvor definierten Alias-basierten URLs funktionieren jedoch nicht mehr.

```
New-IAMAccountAlias -AccountAlias mycompanyaws
```

- Einzelheiten zur API finden Sie unter CreateAccountAlias AWS Tools for PowerShellCmdlet-Referenz.

`New-IAMGroup`

Das folgende Codebeispiel zeigt die Verwendung. `New-IAMGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine neue IAM-Gruppe mit dem Namen **`Developers`** erstellt.

```
New-IAMGroup -GroupName Developers
```

Ausgabe:

```
Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 4/14/2015 11:21:31 AM
GroupId    : QNEJ5PM4NFSQCEXAMPLE1
GroupName  : Developers
Path       : /
```

- Einzelheiten zur API finden Sie unter CreateGroup AWS Tools for PowerShellCmdlet-Referenz.

`New-IAMInstanceProfile`

Das folgende Codebeispiel zeigt die Verwendung. `New-IAMInstanceProfile`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein neues IAM-Instanzprofil mit dem Namen **`ProfileForDevEC2Instance`** erstellt. Sie müssen den **`Add-IAMRoleToInstanceProfile`** Befehl separat ausführen, um das Instanzprofil einer vorhandenen IAM-Rolle zuzuordnen, die Berechtigungen für die Instanz bereitstellt. Fügen Sie abschließend das Instance-Profil einer EC2-Instance bei, wenn Sie sie starten. Verwenden Sie dazu das **`New-EC2Instance`** Cmdlet mit dem Parameter oder. **`InstanceProfile_Arn`** **`InstanceProfile_Name`**

```
New-IAMInstanceProfile -InstanceProfileName ProfileForDevEC2Instance
```

Ausgabe:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/
ProfileForDevEC2Instance
CreateDate          : 4/14/2015 11:31:39 AM
InstanceProfileId   : DYMFXL556EY46EXAMPLE1
InstanceProfileName : ProfileForDevEC2Instance
Path                : /
Roles               : {}
```

- Einzelheiten zur API finden Sie unter CreateInstanceProfile AWS Tools for PowerShellCmdlet-Referenz.

`New-IAMLoginProfile`

Das folgende Codebeispiel zeigt die Verwendung. `New-IAMLoginProfile`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein (temporäres) Passwort für den IAM-Benutzer namens Bob erstellt. Außerdem wird das Kennzeichen gesetzt, dass der Benutzer das Passwort bei der nächsten Anmeldung ändern **Bob** muss.

```
New-IAMLoginProfile -UserName Bob -Password P@ssw0rd -PasswordResetRequired $true
```

Ausgabe:

```
CreateDate                  PasswordResetRequired               UserName
----------                  ---------------------               --------
4/14/2015 12:26:30 PM       True                                Bob
```

- Einzelheiten zur API finden Sie unter CreateLoginProfile AWS Tools for PowerShellCmdlet-Referenz.

`New-IAMOpenIDConnectProvider`

Das folgende Codebeispiel zeigt die Verwendung. `New-IAMOpenIDConnectProvider`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein IAM-OIDC-Anbieter erstellt, der dem OIDC-kompatiblen Anbieterdienst zugeordnet ist, der sich unter der URL **`https://example.oidcprovider.com`**

und der Client-ID befindet. **my-testapp-1** Der OIDC-Anbieter stellt den Fingerabdruck bereit. Um den Fingerabdruck zu authentifizieren, folgen Sie den Schritten unter http://docs.aws.amazon.com/IAM/latest//-thumbprint.html. UserGuide identity-providers-oidc-obtain

```
New-IAMOpenIDConnectProvider -Url https://example.oidcprovider.com -ClientIDList my-
testapp-1 -ThumbprintList 990F419EXAMPLEECF12DDEDA5EXAMPLE52F20D9E
```

Ausgabe:

```
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- Einzelheiten zur API finden Sie unter CreateOpenIdConnectProviderCmdlet-Referenz.AWS Tools for PowerShell

New-IAMPolicy

Das folgende Codebeispiel zeigt die Verwendung. New-IAMPolicy

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird im aktuellen AWS Konto eine neue IAM-Richtlinie mit dem Namen **MySamplePolicy** Die Datei **MySamplePolicy.json** enthält den Richtlinieninhalt erstellt. Beachten Sie, dass Sie den **-Raw** Switch-Parameter verwenden müssen, um die JSON-Richtliniendatei erfolgreich zu verarbeiten.

```
New-IAMPolicy -PolicyName MySamplePolicy -PolicyDocument (Get-Content -Raw
 MySamplePolicy.json)
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter CreatePolicy AWS Tools for PowerShellCmdlet-Referenz.

`New-IAMPolicyVersion`

Das folgende Codebeispiel zeigt die Verwendung. `New-IAMPolicyVersion`

Tools für PowerShell

Beispiel 1: Dieses Beispiel erstellt eine neue „v2“ -Version der IAM-Richtlinie, deren ARN lautet, **`arn:aws:iam::123456789012:policy/MyPolicy`** und macht sie zur Standardversion. Die **`NewPolicyVersion.json`** Datei enthält den Inhalt der Richtlinie. Beachten Sie, dass Sie den **`-Raw`** Switch-Parameter verwenden müssen, um die JSON-Richtliniendatei erfolgreich zu verarbeiten.

```
New-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -
PolicyDocument (Get-content -Raw NewPolicyVersion.json) -SetAsDefault $true
```

Ausgabe:

```
CreateDate                          Document                 IsDefaultVersion
        VersionId
----------                          --------                 ----------------
        ---------
4/15/2015 10:54:54 AM                                        True
        v2
```

- Einzelheiten zur API finden Sie unter CreatePolicyVersion AWS Tools for PowerShellCmdlet-Referenz.

`New-IAMRole`

Das folgende Codebeispiel zeigt die Verwendung. `New-IAMRole`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine neue Rolle mit dem Namen erstellt **`MyNewRole`** und ihr die in der Datei **`NewRoleTrustPolicy.json`** enthaltene Richtlinie angehängt. Beachten

Sie, dass Sie den **`-Raw`** Switch-Parameter verwenden müssen, um die JSON-Richtliniendatei erfolgreich zu verarbeiten. Das in der Ausgabe angezeigte Richtliniendokument ist URL-codiert. In diesem Beispiel wird es mit der **`UrlDecode`** .NET-Methode dekodiert.

```
$results = New-IAMRole -AssumeRolePolicyDocument (Get-Content -raw
 NewRoleTrustPolicy.json) -RoleName MyNewRole
$results
```

Ausgabe:

```
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"
    }
  ]
```

```
}
```

- Einzelheiten zur API finden Sie unter CreateRole AWS Tools for PowerShellCmdlet-Referenz.

`New-IAMSAMLProvider`

Das folgende Codebeispiel zeigt die Verwendung. `New-IAMSAMLProvider`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine neue SAML-Provider-Entität in IAM erstellt. Sie ist benannt **`MySAMLProvider`** und wird durch das SAML-Metadatendokument in der Datei beschrieben**`SAMLMetaData.xml`**, das separat von der Website des SAML-Dienstanbieters heruntergeladen wurde.

```
New-IAMSAMLProvider -Name MySAMLProvider -SAMLMetadataDocument (Get-Content -Raw
 SAMLMetaData.xml)
```

Ausgabe:

```
arn:aws:iam::123456789012:saml-provider/MySAMLProvider
```

- API-Details finden Sie unter CreateSAMLProvider in der Cmdlet-Referenz.AWS Tools for PowerShell

`New-IAMServiceLinkedRole`

Das folgende Codebeispiel zeigt die Verwendung. `New-IAMServiceLinkedRole`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine servicelinkte Rolle für den Autoscaling-Service erstellt.

```
New-IAMServiceLinkedRole -AWSServiceName autoscaling.amazonaws.com -CustomSuffix
 RoleNameEndsWithThis -Description "My service-linked role to support autoscaling"
```

- Einzelheiten zur API finden Sie unter CreateServiceLinkedRoleCmdlet-Referenz.AWS Tools for PowerShell

`New-IAMUser`

Das folgende Codebeispiel zeigt die Verwendung. `New-IAMUser`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein IAM-Benutzer mit dem Namen **`Bob`** erstellt. Wenn Bob sich an der AWS Konsole anmelden muss, müssen Sie den Befehl separat ausführen, **`New-IAMLoginProfile`** um ein Anmeldeprofil mit einem Passwort zu erstellen. Wenn Bob plattformübergreifende CLI-Befehle ausführen AWS PowerShell oder AWS API-Aufrufe tätigen muss, müssen Sie den **`New-IAMAccessKey`** Befehl separat ausführen, um Zugriffsschlüssel zu erstellen.

```
New-IAMUser -UserName Bob
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter CreateUser AWS Tools for PowerShellCmdlet-Referenz.

`New-IAMVirtualMFADevice`

Das folgende Codebeispiel zeigt die Verwendung. `New-IAMVirtualMFADevice`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein neues virtuelles MFA-Gerät erstellt. Die Zeilen 2 und 3 extrahieren den **`Base32StringSeed`** Wert, den das virtuelle MFA-Softwareprogramm benötigt, um ein Konto zu erstellen (als Alternative zum QR-Code). Nachdem Sie das Programm mit dem Wert konfiguriert haben, rufen Sie zwei aufeinanderfolgende Authentifizierungscodes aus dem Programm ab. Verwenden Sie abschließend den letzten Befehl, um das virtuelle MFA-Gerät mit dem IAM-Benutzer zu verknüpfen **`Bob`** und das Konto mit den beiden Authentifizierungscodes zu synchronisieren.

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$SR = New-Object System.IO.StreamReader($Device.Base32StringSeed)
$base32stringseed = $SR.ReadToEnd()
$base32stringseed
CZWZMCQNW4DEXAMPLE3VOUGXJFZYSUW7EXAMPLECR4NJFD65GX2SLUDW2EXAMPLE
```

Ausgabe:

```
-- 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
```

Beispiel 2: In diesem Beispiel wird ein neues virtuelles MFA-Gerät erstellt. Die Zeilen 2 und 3 extrahieren den **`QRCodePNG`** Wert und schreiben ihn in eine Datei. Dieses Bild kann vom virtuellen MFA-Softwareprogramm gescannt werden, um ein Konto zu erstellen (als Alternative zur manuellen Eingabe des StringSeed Base32-Werts). Nachdem Sie das Konto in Ihrem virtuellen MFA-Programm erstellt haben, rufen Sie zwei sequentielle Authentifizierungscodes ab und geben Sie sie in die letzten Befehle ein, um das virtuelle MFA-Gerät mit dem IAM-Benutzer zu verknüpfen **`Bob`** und das Konto zu synchronisieren.

```
$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
```

Ausgabe:

```
 -- Pause here to scan PNG with virtual MFA program to register account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

- Einzelheiten zur API finden Sie unter CreateVirtualMfaDeviceCmdlet-Referenz.AWS Tools for PowerShell

`Publish-IAMServerCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `Publish-IAMServerCertificate`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein neues Serverzertifikat auf das IAM-Konto hochgeladen. Die Dateien, die den Zertifikatshauptteil, den privaten Schlüssel und (optional) die Zertifikatskette enthalten, müssen alle PEM-codiert sein. Beachten Sie, dass die Parameter den tatsächlichen Inhalt der Dateien und nicht die Dateinamen erfordern. Sie müssen den **`-Raw`** Switch-Parameter verwenden, um den Dateiinhalt erfolgreich zu verarbeiten.

```
Publish-IAMServerCertificate -ServerCertificateName MyTestCert -CertificateBody
 (Get-Content -Raw server.crt) -PrivateKey (Get-Content -Raw server.key)
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter UploadServerCertificate AWS Tools for PowerShellCmdlet-Referenz.

`Publish-IAMSigningCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `Publish-IAMSigningCertificate`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein neues X.509-Signaturzertifikat hochgeladen und es dem IAM-Benutzer mit dem Namen zugeordnet. **`Bob`** Die Datei, die den Zertifikatshauptteil enthält, ist PEM-codiert. Der **`CertificateBody`** Parameter erfordert den tatsächlichen Inhalt der Zertifikatsdatei und nicht den Dateinamen. Sie müssen den **`-Raw`** Switch-Parameter verwenden, um die Datei erfolgreich zu verarbeiten.

```
Publish-IAMSigningCertificate -UserName Bob -CertificateBody (Get-Content -Raw
 SampleSigningCert.pem)
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter [UploadSigningCertificate AWS Tools for PowerShell](Cmdlet-Referenz).

`Register-IAMGroupPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Register-IAMGroupPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die vom Kunden verwaltete Richtlinie mit dem Namen **`TesterPolicy`** der IAM-Gruppe verknüpft. **`Testers`** Die Benutzer in dieser Gruppe sind unmittelbar von den in der Standardversion dieser Richtlinie definierten Berechtigungen betroffen.

```
Register-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterPolicy
```

Beispiel 2: In diesem Beispiel wird die AWS verwaltete Richtlinie mit dem Namen **`AdministratorAccess`** der IAM-Gruppe angehängt. **`Admins`** Die Benutzer in dieser Gruppe sind unmittelbar von den in der neuesten Version dieser Richtlinie definierten Berechtigungen betroffen.

```
Register-IAMGroupPolicy -GroupName Admins -PolicyArn arn:aws:iam::aws:policy/
AdministratorAccess
```

- Einzelheiten zur API finden Sie unter AttachGroupPolicy AWS Tools for PowerShellCmdlet-Referenz.

`Register-IAMRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Register-IAMRolePolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die AWS verwaltete Richtlinie mit dem Namen **`SecurityAudit`** der IAM-Rolle angehängt. **`CoSecurityAuditors`** Die Benutzer, die diese Rolle übernehmen, sind unmittelbar von den in der neuesten Version dieser Richtlinie definierten Berechtigungen betroffen.

```
Register-IAMRolePolicy -RoleName CoSecurityAuditors -PolicyArn
 arn:aws:iam::aws:policy/SecurityAudit
```

- Einzelheiten zur API finden Sie unter AttachRolePolicy AWS Tools for PowerShellCmdlet-Referenz.

`Register-IAMUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Register-IAMUserPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die AWS verwaltete Richtlinie mit dem Namen **`AmazonCognitoPowerUser`** des IAM-Benutzers angehängt. **`Bob`** Der Benutzer ist unmittelbar von den in der neuesten Version dieser Richtlinie definierten Berechtigungen betroffen.

```
Register-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::aws:policy/
AmazonCognitoPowerUser
```

- Einzelheiten zur API finden Sie unter AttachUserPolicy AWS Tools for PowerShellCmdlet-Referenz.

`Remove-IAMAccessKey`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMAccessKey`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das AWS Zugriffsschlüsselpaar mit der Schlüssel-ID des **`AKIAIOSFODNN7EXAMPLE`** angegebenen Benutzers gelöscht. **`Bob`**

```
Remove-IAMAccessKey -AccessKeyId AKIAIOSFODNN7EXAMPLE -UserName Bob -Force
```

- Einzelheiten zur API finden Sie unter DeleteAccessKey AWS Tools for PowerShellCmdlet-Referenz.

`Remove-IAMAccountAlias`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMAccountAlias`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der Account-Alias aus Ihrem entfernt AWS-Konto. Die Benutzeranmeldeseite mit dem Alias unter https://mycompanyaws.signin.aws.amazon.com/console funktioniert nicht mehr. Sie müssen stattdessen die ursprüngliche URL mit Ihrer AWS-Konto ID-Nummer unter https://.signin.aws.amazon.com/console verwenden. <accountidnumber>

```
Remove-IAMAccountAlias -AccountAlias mycompanyaws
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz. DeleteAccountAliasAWS Tools for PowerShell

`Remove-IAMAccountPasswordPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMAccountPasswordPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Kennwortrichtlinie für gelöscht AWS-Konto und alle Werte auf ihre ursprünglichen Standardwerte zurückgesetzt. Wenn derzeit keine Kennwortrichtlinie existiert, wird die folgende Fehlermeldung angezeigt: Die Kontorichtlinie mit dem Namen PasswordPolicy kann nicht gefunden werden.

```
Remove-IAMAccountPasswordPolicy
```

- Einzelheiten zur API finden Sie unter DeleteAccountPasswordPolicy AWS Tools for PowerShellCmdlet-Referenz.

`Remove-IAMClientIDFromOpenIDConnectProvider`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMClientIDFromOpenIDConnectProvider`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Client-ID **`My-TestApp-3`** aus der Liste der Client-IDs entfernt, die dem IAM-OIDC-Anbieter zugeordnet sind, dessen ARN lautet. **`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
```

- Einzelheiten zur API finden Sie unter RemoveClientIdFromOpenIdConnectProviderCmdlet-Referenz.AWS Tools for PowerShell

`Remove-IAMGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die IAM-Gruppe mit dem Namen gelöscht. **`MyTestGroup`** Mit dem ersten Befehl werden alle IAM-Benutzer entfernt, die Mitglieder der Gruppe sind, und mit dem zweiten Befehl wird die IAM-Gruppe gelöscht. Beide Befehle funktionieren ohne Aufforderung zur Bestätigung.

```
(Get-IAMGroup -GroupName MyTestGroup).Users | Remove-IAMUserFromGroup -GroupName
 MyTestGroup -Force
Remove-IAMGroup -GroupName MyTestGroup -Force
```

- Einzelheiten zur API finden Sie unter DeleteGroup AWS Tools for PowerShellCmdlet-Referenz.

`Remove-IAMGroupPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMGroupPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die **`TesterPolicy`** aus der IAM-Gruppe **`Testers`** benannte Inline-Richtlinie entfernt. Die Benutzer in dieser Gruppe verlieren sofort die in dieser Richtlinie definierten Berechtigungen.

```
Remove-IAMGroupPolicy -GroupName Testers -PolicyName TestPolicy
```

- Einzelheiten zur API finden Sie unter DeleteGroupPolicy AWS Tools for PowerShellCmdlet-Referenz.

`Remove-IAMInstanceProfile`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMInstanceProfile`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene EC2-Instanzprofil gelöscht. **`MyAppInstanceProfile`** Mit dem ersten Befehl werden alle Rollen vom Instanzprofil getrennt, und mit dem zweiten Befehl wird das Instanzprofil gelöscht.

```
(Get-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile).Roles | Remove-
IAMRoleFromInstanceProfile -InstanceProfileName MyAppInstanceProfile
Remove-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile
```

- Einzelheiten zur API finden Sie unter DeleteInstanceProfile AWS Tools for PowerShellCmdlet-Referenz.

`Remove-IAMLoginProfile`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMLoginProfile`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Anmeldeprofil des IAM-Benutzers mit dem Namen gelöscht. **`Bob`** Dadurch wird verhindert, dass sich der Benutzer an der Konsole anmeldet.

AWS Es verhindert nicht, dass der Benutzer AWS CLI- PowerShell oder API-Aufrufe mit AWS Zugriffsschlüsseln ausführt, die möglicherweise noch mit dem Benutzerkonto verknüpft sind.

```
Remove-IAMLoginProfile -UserName Bob
```

- Einzelheiten zur API finden Sie unter DeleteLoginProfile AWS Tools for PowerShellCmdlet-Referenz.

`Remove-IAMOpenIDConnectProvider`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMOpenIDConnectProvider`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der IAM-OIDC-Anbieter gelöscht, der eine Verbindung zum Anbieter herstellt. **`example.oidcprovider.com`** Stellen Sie sicher, dass Sie alle Rollen aktualisieren oder löschen, die im **`Principal`** Element der Vertrauensrichtlinie der Rolle auf diesen Anbieter verweisen.

```
Remove-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- Einzelheiten zur API finden Sie unter DeleteOpenIdConnectProvider AWS Tools for PowerShellCmdlet-Referenz.

`Remove-IAMPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Richtlinie gelöscht, deren ARN lautet**`arn:aws:iam::123456789012:policy/MySamplePolicy`**. Bevor Sie die Richtlinie löschen können, müssen Sie zuerst alle Versionen mit Ausnahme der Standardversion löschen, indem Sie Folgendes ausführen**`Remove-IAMPolicyVersion`**. Sie müssen die Richtlinie auch von allen IAM-Benutzern, -Gruppen oder -Rollen trennen.

```
Remove-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

Beispiel 2: In diesem Beispiel wird eine Richtlinie gelöscht, indem zuerst alle nicht standardmäßigen Richtlinienversionen gelöscht werden, sie von allen angehängten IAM-Entitäten getrennt und schließlich die Richtlinie selbst gelöscht wird. In der ersten Zeile wird das Richtlinienobjekt abgerufen. In der zweiten Zeile werden alle Richtlinienversionen, die nicht als Standardversion gekennzeichnet sind, in einer Sammlung abgerufen und anschließend alle Richtlinien in der Sammlung gelöscht. In der dritten Zeile werden alle IAM-Benutzer, -Gruppen und -Rollen abgerufen, denen die Richtlinie zugeordnet ist. In den Zeilen vier bis sechs wird die Richtlinie von jeder angehängten Entität getrennt. In der letzten Zeile wird dieser Befehl verwendet, um die verwaltete Richtlinie sowie die verbleibende Standardversion zu entfernen. Das Beispiel enthält den **`-Force`** Switch-Parameter in jeder Zeile, die ihn benötigt, um Bestätigungsaufforderungen zu unterdrücken.

```
$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
```

- Einzelheiten zur API finden Sie unter DeletePolicy AWS Tools for PowerShellCmdlet-Referenz.

`Remove-IAMPolicyVersion`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMPolicyVersion`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die als identifizierte Version **`v2`** aus der Richtlinie gelöscht, deren ARN lautet**`arn:aws:iam::123456789012:policy/MySamplePolicy`**.

```
Remove-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy -
VersionID v2
```

Beispiel 2: In diesem Beispiel wird eine Richtlinie gelöscht, indem zuerst alle nicht standardmäßigen Richtlinienversionen und dann die Richtlinie selbst gelöscht werden.
In der ersten Zeile wird das Richtlinienobjekt abgerufen. In der zweiten Zeile werden alle Richtlinienversionen, die nicht als Standard gekennzeichnet sind, in einer Sammlung abgerufen

und anschließend mit diesem Befehl alle Richtlinien in der Sammlung gelöscht. In der letzten Zeile werden die Richtlinie selbst sowie die verbleibende Standardversion entfernt. Beachten Sie, dass Sie zum erfolgreichen Löschen einer verwalteten Richtlinie auch die Richtlinie mithilfe der **`Unregister-IAMRolePolicy`** Befehle, und von allen Benutzern, Gruppen oder Rollen trennen müssen. **`Unregister-IAMUserPolicy`** **`Unregister-IAMGroupPolicy`** Sehen Sie sich das Beispiel für das **`Remove-IAMPolicy`** Cmdlet an.

```
$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
```

- Einzelheiten zur API finden Sie unter DeletePolicyVersion AWS Tools for PowerShellCmdlet-Referenz.

`Remove-IAMRole`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMRole`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Rolle **`MyNewRole`** aus dem aktuellen IAM-Konto gelöscht. Bevor Sie die Rolle löschen können, müssen Sie zunächst den **`Unregister-IAMRolePolicy`** Befehl verwenden, um alle verwalteten Richtlinien zu trennen. Inline-Richtlinien werden zusammen mit der Rolle gelöscht.

```
Remove-IAMRole -RoleName MyNewRole
```

Beispiel 2: In diesem Beispiel werden alle verwalteten Richtlinien von der genannten Rolle getrennt **`MyNewRole`** und anschließend die Rolle gelöscht. In der ersten Zeile werden alle verwalteten Richtlinien, die der Rolle zugeordnet sind, als Sammlung abgerufen und anschließend jede Richtlinie in der Sammlung von der Rolle getrennt. In der zweiten Zeile wird die Rolle selbst gelöscht. Inline-Richtlinien werden zusammen mit der Rolle gelöscht.

```
Get-IAMAttachedRolePolicyList -RoleName MyNewRole | Unregister-IAMRolePolicy -
RoleName MyNewRole
Remove-IAMRole -RoleName MyNewRole
```

- Einzelheiten zur API finden Sie unter DeleteRole AWS Tools for PowerShellCmdlet-Referenz.

`Remove-IAMRoleFromInstanceProfile`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMRoleFromInstanceProfile`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Rolle **`MyNewRole`** aus dem genannten EC2-Instanzprofil gelöscht. **`MyNewRole`** Ein Instanzprofil, das in der IAM-Konsole erstellt wird, hat immer denselben Namen wie die Rolle, wie in diesem Beispiel. Wenn Sie sie in der API oder CLI erstellen, können sie unterschiedliche Namen haben.

```
Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyNewRole -RoleName MyNewRole
 -Force
```

- Einzelheiten zur API finden Sie unter RemoveRoleFromInstanceProfile AWS Tools for PowerShellCmdlet-Referenz.

`Remove-IAMRolePermissionsBoundary`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMRolePermissionsBoundary`

Tools für PowerShell

Beispiel 1: Dieses Beispiel zeigt, wie die mit einer IAM-Rolle verbundene Berechtigungsgrenze entfernt wird.

```
Remove-IAMRolePermissionsBoundary -RoleName MyRoleName
```

- Einzelheiten zur API finden Sie unter DeleteRolePermissionsBoundary AWS Tools for PowerShellCmdlet-Referenz.

`Remove-IAMRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMRolePolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Inline-Richtlinie gelöscht**`S3AccessPolicy`**, die in die IAM-Rolle eingebettet ist. **`S3BackupRole`**

```
Remove-IAMRolePolicy -PolicyName S3AccessPolicy -RoleName S3BackupRole
```

- Einzelheiten zur API finden Sie unter DeleteRolePolicy AWS Tools for PowerShellCmdlet-Referenz.

`Remove-IAMRoleTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMRoleTag`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Tag aus der Rolle mit dem Namen "MyRoleName" mit dem Tag-Schlüssel „abac“ entfernt. Um mehrere Tags zu entfernen, stellen Sie eine durch Kommas getrennte Tag-Schlüsselliste bereit.

```
Remove-IAMRoleTag -RoleName MyRoleName -TagKey "abac","xyzw"
```

- Einzelheiten zur API finden Sie unter UntagRole AWS Tools for PowerShellCmdlet-Referenz.

`Remove-IAMSAMLProvider`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMSAMLProvider`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der IAM SAML 2.0-Anbieter gelöscht, dessen ARN lautet. **`arn:aws:iam::123456789012:saml-provider/SAMLADFSProvider`**

```
Remove-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFSProvider
```

- API-Einzelheiten finden Sie unter DeleteSAMLProvider in der Cmdlet-Referenz.AWS Tools for PowerShell

`Remove-IAMServerCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMServerCertificate`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Serverzertifikat mit dem Namen **MyServerCert** gelöscht.

```
Remove-IAMServerCertificate -ServerCertificateName MyServerCert
```

- Einzelheiten zur API finden Sie unter DeleteServerCertificate AWS Tools for PowerShellCmdlet-Referenz.

Remove-IAMServiceLinkedRole

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMServiceLinkedRole`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wurde die mit dem Dienst verknüpfte Rolle gelöscht. Bitte beachten Sie, dass dieser Befehl zu einem Fehler führt, wenn der Dienst diese Rolle immer noch verwendet.

```
Remove-IAMServiceLinkedRole -RoleName
 AWSServiceRoleForAutoScaling_RoleNameEndsWithThis
```

- Einzelheiten zur API finden Sie unter DeleteServiceLinkedRole AWS Tools for PowerShellCmdlet-Referenz.

Remove-IAMSigningCertificate

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMSigningCertificate`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Signaturzertifikat mit der ID des IAM-Benutzers mit **Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU** dem Namen gelöscht. **Bob**

```
Remove-IAMSigningCertificate -UserName Bob -CertificateId
 Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
```

- Einzelheiten zur API finden Sie unter DeleteSigningCertificate AWS Tools for PowerShellCmdlet-Referenz.

`Remove-IAMUser`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMUser`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der IAM-Benutzer mit dem Namen gelöscht. **`Bob`**

```
Remove-IAMUser -UserName Bob
```

Beispiel 2: In diesem Beispiel wird der angegebene IAM-Benutzer **`Theresa`** zusammen mit allen Elementen gelöscht, die zuerst gelöscht werden müssen.

```
$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
```

- Einzelheiten zur API finden Sie unter DeleteUser AWS Tools for PowerShellCmdlet-Referenz.

`Remove-IAMUserFromGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMUserFromGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der IAM-Benutzer **`Bob`** aus der Gruppe **`Testers`** entfernt.

```
Remove-IAMUserFromGroup -GroupName Testers -UserName Bob
```

Beispiel 2: In diesem Beispiel werden alle Gruppen gefunden, in denen der IAM-Benutzer Mitglied **`Theresa`** ist, und entfernt sie dann **`Theresa`** aus diesen Gruppen.

```
$groups = Get-IAMGroupForUser -UserName Theresa
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName -
UserName Theresa -Force }
```

Beispiel 3: Dieses Beispiel zeigt eine alternative Möglichkeit, den IAM-Benutzer **`Bob`** aus der **`Testers`** Gruppe zu entfernen.

```
Get-IAMGroupForUser -UserName Bob | Remove-IAMUserFromGroup -UserName Bob -GroupName
 Testers -Force
```

- Einzelheiten zur API finden Sie unter RemoveUserFromGroup AWS Tools for PowerShellCmdlet-Referenz.

`Remove-IAMUserPermissionsBoundary`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMUserPermissionsBoundary`

Tools für PowerShell

Beispiel 1: Dieses Beispiel zeigt, wie die einem IAM-Benutzer zugeordnete Berechtigungsgrenze entfernt wird.

```
Remove-IAMUserPermissionsBoundary -UserName joe
```

- Einzelheiten zur API finden Sie unter DeleteUserPermissionsBoundary AWS Tools for PowerShellCmdlet-Referenz.

`Remove-IAMUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMUserPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Inline-Richtlinie gelöscht**`AccessToEC2Policy`**, die in den IAM-Benutzer mit dem Namen eingebettet ist. **`Bob`**

```
Remove-IAMUserPolicy -PolicyName AccessToEC2Policy -UserName Bob
```

Beispiel 2: In diesem Beispiel werden alle Inline-Richtlinien gefunden, die in den IAM-Benutzernamen eingebettet sind, **`Theresa`** und sie werden dann gelöscht.

```
$inlinepols = Get-IAMUserPolicies -UserName Theresa
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
 Theresa -Force}
```

- Einzelheiten zur API finden Sie unter DeleteUserPolicyCmdlet-Referenz.AWS Tools for PowerShell

`Remove-IAMUserTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMUserTag`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Tag vom Benutzer mit dem Namen „joe“ und dem Tag-Schlüssel „abac“ und „xyzw“ entfernt. Um mehrere Tags zu entfernen, geben Sie eine durch Kommas getrennte Liste der Tag-Schlüssel an.

```
Remove-IAMUserTag -UserName joe -TagKey "abac","xyzw"
```

- Einzelheiten zur API finden Sie unter UntagUser AWS Tools for PowerShellCmdlet-Referenz.

`Remove-IAMVirtualMFADevice`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMVirtualMFADevice`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das virtuelle IAM-MFA-Gerät gelöscht, dessen ARN lautet. **`arn:aws:iam::123456789012:mfa/bob`**

```
Remove-IAMVirtualMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/bob
```

Beispiel 2: In diesem Beispiel wird geprüft, ob der IAM-Benutzerin Theresa ein MFA-Gerät zugewiesen wurde. Wenn eines gefunden wird, ist das Gerät für den IAM-Benutzer deaktiviert. Wenn das Gerät virtuell ist, wird es ebenfalls gelöscht.

```
$mfa = Get-IAMMFADevice -UserName Theresa
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -SerialNumber
 $mfa.SerialNumber }
}
```

- Einzelheiten zur API finden Sie unter DeleteVirtualMfaDevice AWS Tools for PowerShellCmdlet-Referenz.

`Request-IAMCredentialReport`

Das folgende Codebeispiel zeigt die Verwendung. `Request-IAMCredentialReport`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Generierung eines neuen Berichts angefordert, was alle vier Stunden erfolgen kann. Wenn der letzte Bericht noch aktuell ist, lautet das Feld Bundesland**COMPLETE**. **Get-IAMCredentialReport**Dient zum Anzeigen des abgeschlossenen Berichts.

```
Request-IAMCredentialReport
```

Ausgabe:

```
Description                                                  State
-----------                                                  -----
No report exists. Starting a new report generation task      STARTED
```

- Einzelheiten zur API finden Sie unter GenerateCredentialReport AWS Tools for PowerShellCmdlet-Referenz.

Request-IAMServiceLastAccessedDetail

Das folgende Codebeispiel zeigt die Verwendung. `Request-IAMServiceLastAccessedDetail`

Tools für PowerShell

Beispiel 1: Dieses Beispiel entspricht einem API-Cmdlet. GenerateServiceLastAccessedDetails Dies bietet eine Job-ID, die in Get-IAM und ServiceLastAccessedDetail Get-IAM verwendet werden kann ServiceLastAccessedDetailWithEntity

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

- Einzelheiten zur API finden Sie unter GenerateServiceLastAccessedDetailsCmdlet-Referenz.AWS Tools for PowerShell

Set-IAMDefaultPolicyVersion

Das folgende Codebeispiel zeigt die Verwendung. `Set-IAMDefaultPolicyVersion`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die **v2** Version der Richtlinie festgelegt, deren ARN die aktive Standardversion ist**arn:aws:iam::123456789012:policy/MyPolicy**.

```
Set-IAMDefaultPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -
VersionId v2
```

- Einzelheiten zur API finden Sie unter SetDefaultPolicyVersion AWS Tools for PowerShellCmdlet-Referenz.

`Set-IAMRolePermissionsBoundary`

Das folgende Codebeispiel zeigt die Verwendung. `Set-IAMRolePermissionsBoundary`

Tools für PowerShell

Beispiel 1: Dieses Beispiel zeigt, wie die Berechtigungsgrenze für eine IAM-Rolle festgelegt wird. Sie können AWS verwaltete Richtlinien oder benutzerdefinierte Richtlinien als Berechtigungsgrenze festlegen.

```
Set-IAMRolePermissionsBoundary -RoleName MyRoleName -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- Einzelheiten zur API finden Sie unter PutRolePermissionsBoundary AWS Tools for PowerShellCmdlet-Referenz.

`Set-IAMUserPermissionsBoundary`

Das folgende Codebeispiel zeigt die Verwendung. `Set-IAMUserPermissionsBoundary`

Tools für PowerShell

Beispiel 1: Dieses Beispiel zeigt, wie die Berechtigungsgrenze für den Benutzer festgelegt wird. Sie können AWS verwaltete Richtlinien oder benutzerdefinierte Richtlinien als Berechtigungsgrenze festlegen.

```
Set-IAMUserPermissionsBoundary -UserName joe -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- Einzelheiten zur API finden Sie unter PutUserPermissionsBoundary AWS Tools for PowerShellCmdlet-Referenz.

`Sync-IAMMFADevice`

Das folgende Codebeispiel zeigt die Verwendung. `Sync-IAMMFADevice`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das MFA-Gerät synchronisiert, das dem IAM-Benutzer zugeordnet ist **`Bob`** und dessen ARN **`arn:aws:iam::123456789012:mfa/bob`** mit einem Authentifizierungsprogramm verknüpft ist, das die beiden Authentifizierungscodes bereitgestellt hat.

```
Sync-IAMMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/theresa -
AuthenticationCode1 123456 -AuthenticationCode2 987654 -UserName Bob
```

Beispiel 2: In diesem Beispiel wird das IAM-MFA-Gerät, das dem IAM-Benutzer zugeordnet ist, **`Theresa`** mit einem physischen Gerät synchronisiert, das die Seriennummer hat **`ABCD12345678`** und das die beiden Authentifizierungscodes bereitgestellt hat.

```
Sync-IAMMFADevice -SerialNumber ABCD12345678 -AuthenticationCode1 123456 -
AuthenticationCode2 987654 -UserName Theresa
```

- Einzelheiten zur API finden Sie unter ResyncMfaDeviceCmdlet-Referenz.AWS Tools for PowerShell

`Unregister-IAMGroupPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-IAMGroupPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die verwaltete Gruppenrichtlinie, deren ARN stammt, **`arn:aws:iam::123456789012:policy/TesterAccessPolicy`** von der genannten **`Testers`** Gruppe getrennt.

```
Unregister-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterAccessPolicy
```

Beispiel 2: In diesem Beispiel werden alle verwalteten Richtlinien gefunden, die der genannten Gruppe zugeordnet sind, **`Testers`** und sie werden von der Gruppe getrennt.

```
Get-IAMAttachedGroupPolicies -GroupName Testers | Unregister-IAMGroupPolicy -
Groupname Testers
```

- Einzelheiten zur API finden Sie unter DetachGroupPolicy AWS Tools for PowerShellCmdlet-Referenz.

`Unregister-IAMRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-IAMRolePolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die verwaltete Gruppenrichtlinie, deren ARN stammt, **`arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy`** von der genannten **`FedTesterRole`** Rolle getrennt.

```
Unregister-IAMRolePolicy -RoleName FedTesterRole -PolicyArn
 arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy
```

Beispiel 2: In diesem Beispiel werden alle verwalteten Richtlinien gefunden, die der genannten Rolle zugeordnet sind, **`FedTesterRole`** und sie werden von der Rolle getrennt.

```
Get-IAMAttachedRolePolicyList -RoleName FedTesterRole | Unregister-IAMRolePolicy -
Rolename FedTesterRole
```

- Einzelheiten zur API finden Sie unter DetachRolePolicy AWS Tools for PowerShellCmdlet-Referenz.

`Unregister-IAMUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-IAMUserPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die verwaltete Richtlinie, deren ARN stammt, **`arn:aws:iam::123456789012:policy/TesterPolicy`** von dem IAM-Benutzer mit dem Namen getrennt. **`Bob`**

```
Unregister-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::123456789012:policy/
TesterPolicy
```

Beispiel 2: In diesem Beispiel werden alle verwalteten Richtlinien gefunden, die dem IAM-Benutzer mit dem Namen zugeordnet sind, **`Theresa`** und diese Richtlinien werden vom Benutzer getrennt.

```
Get-IAMAttachedUserPolicyList -UserName Theresa | Unregister-IAMUserPolicy -Username
 Theresa
```

- Einzelheiten zur API finden Sie unter DetachUserPolicy AWS Tools for PowerShellCmdlet-Referenz.

`Update-IAMAccessKey`

Das folgende Codebeispiel zeigt die Verwendung. `Update-IAMAccessKey`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der Status des Zugriffsschlüssels **`AKIAIOSFODNN7EXAMPLE`** für den IAM-Benutzer mit dem Namen **`Bob`** to **`Inactive`** geändert.

```
Update-IAMAccessKey -UserName Bob -AccessKeyId AKIAIOSFODNN7EXAMPLE -Status Inactive
```

- Einzelheiten zur API finden Sie unter UpdateAccessKey AWS Tools for PowerShellCmdlet-Referenz.

`Update-IAMAccountPasswordPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Update-IAMAccountPasswordPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Kennwortrichtlinie für das Konto mit den angegebenen Einstellungen aktualisiert. Beachten Sie, dass alle Parameter, die nicht im Befehl enthalten sind, nicht unverändert bleiben. Stattdessen werden sie auf die Standardwerte zurückgesetzt.

```
Update-IAMAccountPasswordPolicy -AllowUsersToChangePasswords $true -HardExpiry
 $false -MaxPasswordAge 90 -MinimumPasswordLength 8 -PasswordReusePrevention 20
```

```
 -RequireLowercaseCharacters $true -RequireNumbers $true -RequireSymbols $true -
RequireUppercaseCharacters $true
```

- Einzelheiten zur API finden Sie unter UpdateAccountPasswordPolicy AWS Tools for PowerShellCmdlet-Referenz.

`Update-IAMAssumeRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Update-IAMAssumeRolePolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die benannte IAM-Rolle **`ClientRole`** mit einer neuen Vertrauensrichtlinie aktualisiert, deren Inhalt aus der Datei **`ClientRolePolicy.json`** stammt. Beachten Sie, dass Sie den **`-Raw`** Switch-Parameter verwenden müssen, um den Inhalt der JSON-Datei erfolgreich zu verarbeiten.

```
Update-IAMAssumeRolePolicy -RoleName ClientRole -PolicyDocument (Get-Content -raw
 ClientRolePolicy.json)
```

- Einzelheiten zur API finden Sie unter UpdateAssumeRolePolicy AWS Tools for PowerShellCmdlet-Referenz.

`Update-IAMGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Update-IAMGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die IAM-Gruppe **`Testers`** in umbenannt. **`AppTesters`**

```
Update-IAMGroup -GroupName Testers -NewGroupName AppTesters
```

Beispiel 2: In diesem Beispiel wird der Pfad der IAM-Gruppe in geändert. **`AppTesters`** **`/Org1/Org2/`** Dadurch wird der ARN für die Gruppe auf geändert**`arn:aws:iam::123456789012:group/Org1/Org2/AppTesters`**.

```
Update-IAMGroup -GroupName AppTesters -NewPath /Org1/Org2/
```

- Einzelheiten zur API finden Sie unter UpdateGroup AWS Tools for PowerShellCmdlet-Referenz.

`Update-IAMLoginProfile`

Das folgende Codebeispiel zeigt die Verwendung. `Update-IAMLoginProfile`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein neues temporäres Passwort für den IAM-Benutzer festgelegt und der Benutzer muss das Passwort ändern**Bob**, wenn er sich das nächste Mal anmeldet.

```
Update-IAMLoginProfile -UserName Bob -Password "P@ssw0rd1234" -PasswordResetRequired
 $true
```

- Einzelheiten zur API finden Sie unter UpdateLoginProfile AWS Tools for PowerShellCmdlet-Referenz.

`Update-IAMOpenIDConnectProviderThumbprint`

Das folgende Codebeispiel zeigt die Verwendung. `Update-IAMOpenIDConnectProviderThumbprint`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Zertifikat-Fingerabdruckliste für den OIDC-Anbieter aktualisiert, dessen ARN einen neuen Fingerabdruck verwenden **`arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com`** soll. Der OIDC-Anbieter teilt den neuen Wert, wenn sich das Zertifikat, das dem Anbieter zugeordnet ist, ändert.

```
Update-IAMOpenIDConnectProviderThumbprint -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com -ThumbprintList
 7359755EXAMPLEabc3060bce3EXAMPLEec4542a3
```

- Einzelheiten zur API finden Sie unter UpdateOpenIdConnectProviderThumbprint AWS Tools for PowerShellCmdlet-Referenz.

`Update-IAMRole`

Das folgende Codebeispiel zeigt die Verwendung. `Update-IAMRole`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Rollenbeschreibung und der Wert für die maximale Sitzungsdauer (in Sekunden) aktualisiert, für den eine Rollensitzung angefordert werden kann.

```
Update-IAMRole -RoleName MyRoleName -Description "My testing role" -
MaxSessionDuration 43200
```

- Einzelheiten zur API finden Sie unter UpdateRole AWS Tools for PowerShellCmdlet-Referenz.

`Update-IAMRoleDescription`

Das folgende Codebeispiel zeigt die Verwendung. `Update-IAMRoleDescription`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Beschreibung einer IAM-Rolle in Ihrem Konto aktualisiert.

```
Update-IAMRoleDescription -RoleName MyRoleName -Description "My testing role"
```

- Einzelheiten zur API finden Sie unter UpdateRoleDescription AWS Tools for PowerShellCmdlet-Referenz.

`Update-IAMSAMLProvider`

Das folgende Codebeispiel zeigt die Verwendung. `Update-IAMSAMLProvider`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der SAML-Anbieter in IAM, dessen ARN lautet, **`arn:aws:iam::123456789012:saml-provider/SAMLADFS`** mit einem neuen SAML-Metadatendokument aus der Datei aktualisiert. **`SAMLMetaData.xml`** Beachten Sie, dass Sie den **`-Raw`** Switch-Parameter verwenden müssen, um den Inhalt der JSON-Datei erfolgreich zu verarbeiten.

```
Update-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS -SAMLMetadataDocument (Get-Content -Raw SAMLMetaData.xml)
```

- Einzelheiten zur API finden Sie unter UpdateSamlProvider AWS Tools for PowerShellCmdlet-Referenz.

`Update-IAMServerCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `Update-IAMServerCertificate`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Zertifikat mit dem Namen to umbenannt**`MyServerCertificate`**. **`MyRenamedServerCertificate`**

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -
NewServerCertificateName MyRenamedServerCertificate
```

Beispiel 2: In diesem Beispiel wird das Zertifikat mit dem Namen in den Pfad **`MyServerCertificate`** /Org1/Org2/verschoben. Dadurch wird der ARN für die Ressource auf geändert**`arn:aws:iam::123456789012:server-certificate/Org1/Org2/MyServerCertificate`**.

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -NewPath /
Org1/Org2/
```

- Einzelheiten zur API finden Sie unter UpdateServerCertificate AWS Tools for PowerShellCmdlet-Referenz.

`Update-IAMSigningCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `Update-IAMSigningCertificate`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Zertifikat aktualisiert, das dem genannten IAM-Benutzer zugeordnet ist **`Bob`** und dessen Zertifikat-ID es als inaktiv kennzeichnet. **`Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU`**

```
Update-IAMSigningCertificate -CertificateId Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU -
UserName Bob -Status Inactive
```

- Einzelheiten zur API finden Sie unter UpdateSigningCertificate AWS Tools for PowerShellCmdlet-Referenz.

`Update-IAMUser`

Das folgende Codebeispiel zeigt die Verwendung. `Update-IAMUser`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der IAM-Benutzer **`Bob`** in umbenannt. **`Robert`**

```
Update-IAMUser -UserName Bob -NewUserName Robert
```

Beispiel 2: In diesem Beispiel wird der Pfad des IAM-Benutzers **`Bob`** auf geändert**`/Org1/Org2/`**, wodurch der ARN für den Benutzer effektiv geändert wird. **`arn:aws:iam::123456789012:user/Org1/Org2/bob`**

```
Update-IAMUser -UserName Bob -NewPath /Org1/Org2/
```

- Einzelheiten zur API finden Sie unter UpdateUser AWS Tools for PowerShellCmdlet-Referenz.

`Write-IAMGroupPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Write-IAMGroupPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Inline-Richtlinie mit dem Namen erstellt **`AppTesterPolicy`** und in die IAM-Gruppe eingebettet. **`AppTesters`** Wenn bereits eine Inline-Richtlinie mit demselben Namen existiert, wird sie überschrieben. Der Inhalt der JSON-Richtlinie wird in der Datei **`apptesterpolicy.json`** gespeichert. Beachten Sie, dass Sie den **`-Raw`** Parameter verwenden müssen, um den Inhalt der JSON-Datei erfolgreich zu verarbeiten.

```
Write-IAMGroupPolicy -GroupName AppTesters -PolicyName AppTesterPolicy -
PolicyDocument (Get-Content -Raw apptesterpolicy.json)
```

- Einzelheiten zur API finden Sie unter PutGroupPolicy AWS Tools for PowerShellCmdlet-Referenz.

`Write-IAMRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Write-IAMRolePolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Inline-Richtlinie mit dem Namen erstellt **`FedTesterRolePolicy`** und in die IAM-Rolle eingebettet. **`FedTesterRole`** Wenn bereits eine Inline-Richtlinie mit demselben Namen existiert, wird sie überschrieben. Der Inhalt der JSON-Richtlinie stammt aus der Datei**`FedTesterPolicy.json`**. Beachten Sie, dass Sie den **`-Raw`** Parameter verwenden müssen, um den Inhalt der JSON-Datei erfolgreich zu verarbeiten.

```
Write-IAMRolePolicy -RoleName FedTesterRole -PolicyName FedTesterRolePolicy -
PolicyDocument (Get-Content -Raw FedTesterPolicy.json)
```

- Einzelheiten zur API finden Sie unter PutRolePolicy AWS Tools for PowerShellCmdlet-Referenz.

`Write-IAMUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Write-IAMUserPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Inline-Richtlinie mit dem Namen erstellt **`EC2AccessPolicy`** und in den IAM-Benutzer eingebettet. **`Bob`** Wenn bereits eine Inline-Richtlinie mit demselben Namen existiert, wird sie überschrieben. Der Inhalt der JSON-Richtlinie stammt aus der Datei**`EC2AccessPolicy.json`**. Beachten Sie, dass Sie den **`-Raw`** Parameter verwenden müssen, um den Inhalt der JSON-Datei erfolgreich zu verarbeiten.

```
Write-IAMUserPolicy -UserName Bob -PolicyName EC2AccessPolicy -PolicyDocument (Get-
Content -Raw EC2AccessPolicy.json)
```

- Einzelheiten zur API finden Sie unter PutUserPolicy AWS Tools for PowerShellCmdlet-Referenz.

Kinesis-Beispiele für die Verwendung von Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS Tools for PowerShell mit Kinesis Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

Aktionen

`Get-KINRecord`

Das folgende Codebeispiel zeigt die Verwendung`Get-KINRecord`.

Tools für PowerShell

Beispiel 1: Dieses Beispiel zeigt, wie Daten aus einer Reihe von einem oder mehreren Datensätzen zurückgegeben und extrahiert werden. Der für Get-KinRecord bereitgestellte Iterator bestimmt die Startposition der zurückzugebenden Datensätze, die in diesem Beispiel in der Variablen $records erfasst werden. Auf jeden einzelnen Datensatz kann dann zugegriffen werden, indem die $records-Auflistung indexiert wird. Unter der Annahme, dass es sich bei den Daten im Datensatz um UTF-8-codierten Text handelt, zeigt der letzte Befehl, wie Sie die Daten aus dem Objekt extrahieren und als Text MemoryStream an die Konsole zurückgeben können.

```
$records
$records = Get-KINRecord -ShardIterator "AAAAAAAAAAGIc....9VnbiRNaP"
```

Ausgabe:

```
MillisBehindLatest NextShardIterator            Records
------------------ -----------------            -------
0                  AAAAAAAAAAERNIq...uDn11HuUs  {Key1, Key2}
```

```
$records.Records[0]
```

Ausgabe:

```
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())
```

Ausgabe:

```
test data from string
```

- Einzelheiten zur API finden Sie unter GetRecordsCmdlet-Referenz.AWS Tools for PowerShell

`Get-KINShardIterator`

Das folgende Codebeispiel zeigt die Verwendung. `Get-KINShardIterator`

Tools für PowerShell

Beispiel 1: Gibt einen Shard-Iterator für den angegebenen Shard und die angegebene Startposition zurück. Einzelheiten zu den Shard-IDs und Sequenznummern können der Ausgabe des Cmdlets Get-KinStream entnommen werden, indem auf die Shards-Auflistung des zurückgegebenen Stream-Objekts verwiesen wird. Der zurückgegebene Iterator kann mit dem Cmdlet Get-KinRecord verwendet werden, um Datensätze im Shard abzurufen.

```
Get-KINShardIterator -StreamName "mystream" -ShardId "shardId-000000000000" -
ShardIteratorType AT_SEQUENCE_NUMBER -StartingSequenceNumber "495598645..."
```

Ausgabe:

```
AAAAAAAAAAGIc....9VnbiRNaP
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz. GetShardIteratorAWS Tools for PowerShell

`Get-KINStream`

Das folgende Codebeispiel zeigt die Verwendung. `Get-KINStream`

Tools für PowerShell

Beispiel 1: Gibt Details des angegebenen Streams zurück.

```
Get-KINStream -StreamName "mystream"
```

Ausgabe:

```
HasMoreShards        : False
RetentionPeriodHours : 24
Shards               : {}
StreamARN            : arn:aws:kinesis:us-west-2:123456789012:stream/mystream
StreamName           : mystream
StreamStatus         : ACTIVE
```

- Einzelheiten zur API finden Sie unter DescribeStream AWS Tools for PowerShellCmdlet-Referenz.

`New-KINStream`

Das folgende Codebeispiel zeigt die Verwendung. `New-KINStream`

Tools für PowerShell

Beispiel 1: Erzeugt einen neuen Stream. Standardmäßig gibt dieses Cmdlet keine Ausgabe zurück. Daher wird der PassThru Schalter - hinzugefügt, um den Wert zurückzugeben, der dem StreamName Parameter - zur späteren Verwendung übergeben wurde.

```
$streamName = New-KINStream -StreamName "mystream" -ShardCount 1 -PassThru
```

- Einzelheiten zur API finden Sie unter CreateStream AWS Tools for PowerShellCmdlet-Referenz.

`Remove-KINStream`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-KINStream`

Tools für PowerShell

Beispiel 1: Löscht den angegebenen Stream. Sie werden zur Bestätigung aufgefordert, bevor der Befehl ausgeführt wird. Verwenden Sie den Schalter -Force, um die Bestätigungsaufforderung zu unterdrücken.

```
Remove-KINStream -StreamName "mystream"
```

- Einzelheiten zur API finden Sie unter DeleteStream AWS Tools for PowerShellCmdlet-Referenz.

`Write-KINRecord`

Das folgende Codebeispiel zeigt die Verwendung. `Write-KINRecord`

Tools für PowerShell

Beispiel 1: Schreibt einen Datensatz, der die im Parameter -Text angegebene Zeichenfolge enthält.

```
Write-KINRecord -Text "test data from string" -StreamName "mystream" -PartitionKey
 "Key1"
```

Beispiel 2: Schreibt einen Datensatz, der die in der angegebenen Datei enthaltenen Daten enthält. Die Datei wird als Bytefolge behandelt. Wenn sie Text enthält, sollte sie mit der erforderlichen Kodierung geschrieben werden, bevor sie mit diesem Cmdlet verwendet wird.

```
Write-KINRecord -FilePath "C:\TestData.txt" -StreamName "mystream" -PartitionKey
 "Key2"
```

- Einzelheiten zur API finden Sie unter PutRecord AWS Tools for PowerShellCmdlet-Referenz.

Lambda-Beispiele mit Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS Tools for PowerShell mit Lambda Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

Aktionen

`Add-LMResourceTag`

Das folgende Codebeispiel zeigt die Verwendung`Add-LMResourceTag`.

Tools für PowerShell

Beispiel 1: Fügt die drei Tags (Washington, Oregon und Kalifornien) und ihre zugehörigen Werte der angegebenen Funktion hinzu, die durch ihren ARN identifiziert wird.

```
Add-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction" -Tag @{ "Washington" = "Olympia"; "Oregon"
 = "Salem"; "California" = "Sacramento" }
```

- Einzelheiten zur API finden Sie unter TagResource AWS Tools for PowerShellCmdlet-Referenz.

`Get-LMAccountSetting`

Das folgende Codebeispiel zeigt die Verwendung. `Get-LMAccountSetting`

Tools für PowerShell

Beispiel 1: Dieses Beispiel wird angezeigt, um das Kontolimit und die Kontonutzung zu vergleichen

```
Get-LMAccountSetting | Select-Object
 @{Name="TotalCodeSizeLimit";Expression={$_.AccountLimit.TotalCodeSize}},
 @{Name="TotalCodeSizeUsed";Expression={$_.AccountUsage.TotalCodeSize}}
```

Ausgabe:

```
TotalCodeSizeLimit TotalCodeSizeUsed
------------------ -----------------
       80530636800          15078795
```

- Einzelheiten zur API finden Sie unter GetAccountSettings AWS Tools for PowerShellCmdlet-Referenz.

`Get-LMAlias`

Das folgende Codebeispiel zeigt die Verwendung. `Get-LMAlias`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Gewichtungen der Routing-Konfiguration für einen bestimmten Lambda-Funktionsalias abgerufen.

```
Get-LMAlias -FunctionName "MylambdaFunction123" -Name "newlabel1" -Select
 RoutingConfig
```

Ausgabe:

```
AdditionalVersionWeights
------------------------
{[1, 0.6]}
```

- Einzelheiten zur API finden Sie unter GetAlias AWS Tools for PowerShellCmdlet-Referenz.

`Get-LMFunctionConcurrency`

Das folgende Codebeispiel zeigt die Verwendung. `Get-LMFunctionConcurrency`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die reservierte Parallelität für die Lambda-Funktion abgerufen

```
Get-LMFunctionConcurrency -FunctionName "MylambdaFunction123" -Select *
```

Ausgabe:

```
ReservedConcurrentExecutions
----------------------------
100
```

- Einzelheiten zur API finden Sie unter GetFunctionConcurrency AWS Tools for PowerShellCmdlet-Referenz.

`Get-LMFunctionConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Get-LMFunctionConfiguration`

Tools für PowerShell

Beispiel 1: Dieses Beispiel gibt die versionsspezifische Konfiguration einer Lambda-Funktion zurück.

```
Get-LMFunctionConfiguration -FunctionName "MylambdaFunction123" -Qualifier
 "PowershellAlias"
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter GetFunctionConfiguration AWS Tools for PowerShellCmdlet-Referenz.

`Get-LMFunctionList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-LMFunctionList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden alle Lambda-Funktionen mit sortierter Codegröße angezeigt

```
Get-LMFunctionList | Sort-Object -Property CodeSize | Select-Object FunctionName,
 RunTime, Timeout, CodeSize
```

Ausgabe:

```
FunctionName                                                Runtime   Timeout
 CodeSize
------------                                                -------   -------
 --------
test                                                        python2.7       3
 243
MylambdaFunction123                                         python3.8     600
 659
myfuncpython1                                               python3.8     303
 675
```

- Einzelheiten zur API finden Sie unter ListFunctions AWS Tools for PowerShellCmdlet-Referenz.

`Get-LMPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Get-LMPolicy`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Funktionsrichtlinie der Lambda-Funktion angezeigt

```
Get-LMPolicy -FunctionName test -Select Policy
```

Ausgabe:

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

- Einzelheiten zur API finden Sie unter GetPolicy AWS Tools for PowerShellCmdlet-Referenz.

`Get-LMProvisionedConcurrencyConfig`

Das folgende Codebeispiel zeigt die Verwendung. `Get-LMProvisionedConcurrencyConfig`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die bereitgestellte Parallelitätskonfiguration für den angegebenen Alias der Lambda-Funktion abgerufen.

```
C:\>Get-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -
Qualifier "NewAlias1"
```

Ausgabe:

```
AllocatedProvisionedConcurrentExecutions : 0
AvailableProvisionedConcurrentExecutions : 0
LastModified                             : 2020-01-15T03:21:26+0000
RequestedProvisionedConcurrentExecutions : 70
Status                                   : IN_PROGRESS
StatusReason                             :
```

- Einzelheiten zur API finden Sie unter GetProvisionedConcurrencyConfigCmdlet-Referenz.AWS Tools for PowerShell

`Get-LMProvisionedConcurrencyConfigList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-LMProvisionedConcurrencyConfigList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Liste der bereitgestellten Parallelitätskonfigurationen für eine Lambda-Funktion abgerufen.

```
Get-LMProvisionedConcurrencyConfigList -FunctionName "MylambdaFunction123"
```

- Einzelheiten zur API finden Sie unter ListProvisionedConcurrencyConfigsCmdlet-Referenz.AWS Tools for PowerShell

`Get-LMResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Get-LMResourceTag`

Tools für PowerShell

Beispiel 1: Ruft die Tags und ihre Werte ab, die derzeit für die angegebene Funktion festgelegt sind.

```
Get-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction"
```

Ausgabe:

```
Key        Value
---        -----
California Sacramento
Oregon     Salem
Washington Olympia
```

- Einzelheiten zur API finden Sie unter ListTags AWS Tools for PowerShellCmdlet-Referenz.

`Get-LMVersionsByFunction`

Das folgende Codebeispiel zeigt die Verwendung. `Get-LMVersionsByFunction`

Tools für PowerShell

Beispiel 1: Dieses Beispiel gibt die Liste der versionsspezifischen Konfigurationen für jede Version der Lambda-Funktion zurück.

```
Get-LMVersionsByFunction -FunctionName "MylambdaFunction123"
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter ListVersionsByFunction AWS Tools for PowerShellCmdlet-Referenz.

`New-LMAlias`

Das folgende Codebeispiel zeigt die Verwendung. `New-LMAlias`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein neuer Lambda-Alias für die angegebene Version und Routing-Konfiguration erstellt, um den Prozentsatz der empfangenen Aufrufanforderungen anzugeben.

```
New-LMAlias -FunctionName "MylambdaFunction123" -
RoutingConfig_AdditionalVersionWeight @{Name="1";Value="0.6} -Description "Alias for
 version 4" -FunctionVersion 4 -Name "PowershellAlias"
```

- Einzelheiten zur API finden Sie unter CreateAlias AWS Tools for PowerShellCmdlet-Referenz.

`Publish-LMFunction`

Das folgende Codebeispiel zeigt die Verwendung. `Publish-LMFunction`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine neue C#-Funktion (dotnetcore1.0 Runtime) mit dem Namen AWS Lambda erstellt, die die kompilierten Binärdateien für die Funktion aus einer ZIP-Datei MyFunction im lokalen Dateisystem bereitstellt (relative oder absolute Pfade können verwendet werden). C#-Lambda-Funktionen spezifizieren den Handler für die Funktion mit der Bezeichnung AssemblyName: :Namespace. ClassName::. MethodName Sie sollten den Assemblynamen (ohne DLL-Suffix), den Namespace, den Klassennamen und den Methodennamen der Handler-Spezifikation entsprechend ersetzen. Für die neue Funktion werden die Umgebungsvariablen 'envvar1' und 'envvar2' aus den bereitgestellten Werten eingerichtet.

```
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" }
```

Ausgabe:

```
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        :
```

Beispiel 2: Dieses Beispiel ähnelt dem vorherigen, außer dass die Funktionsbinärdateien zuerst in einen Amazon S3 S3-Bucket hochgeladen werden (der sich in derselben Region wie die beabsichtigte Lambda-Funktion befinden muss) und das resultierende S3-Objekt dann beim Erstellen der Funktion referenziert wird.

```
Write-S3Object -BucketName mybucket -Key MyFunctionBinaries.zip -File .
\MyFunctionBinaries.zip
Publish-LMFunction -Description "My C# Lambda Function" `
        -FunctionName MyFunction `
        -BucketName mybucket `
        -Key MyFunctionBinaries.zip `
        -Handler "AssemblyName::Namespace.ClassName::MethodName" `
        -Role "arn:aws:iam::123456789012:role/LambdaFullExecRole" `
        -Runtime dotnetcore1.0 `
        -Environment_Variable @{ "envvar1"="value";"envvar2"="value" }
```

- Einzelheiten zur API finden Sie unter CreateFunction AWS Tools for PowerShellCmdlet-Referenz.

`Publish-LMVersion`

Das folgende Codebeispiel zeigt die Verwendung. `Publish-LMVersion`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Version für den vorhandenen Snapshot von Lambda Function Code erstellt

```
Publish-LMVersion -FunctionName "MylambdaFunction123" -Description "Publishing
 Existing Snapshot of function code as a  new version through Powershell"
```

- Einzelheiten zur API finden Sie unter PublishVersion AWS Tools for PowerShellCmdlet-Referenz.

`Remove-LMAlias`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-LMAlias`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die im Befehl erwähnte Lambda-Funktion Alias gelöscht.

```
Remove-LMAlias -FunctionName "MylambdaFunction123" -Name "NewAlias"
```

- Einzelheiten zur API finden Sie unter DeleteAlias AWS Tools for PowerShellCmdlet-Referenz.

`Remove-LMFunction`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-LMFunction`

Tools für PowerShell

Beispiel 1: Dieses Beispiel löscht eine bestimmte Version einer Lambda-Funktion

```
Remove-LMFunction -FunctionName "MylambdaFunction123" -Qualifier '3'
```

- Einzelheiten zur API finden Sie unter DeleteFunction AWS Tools for PowerShellCmdlet-Referenz.

`Remove-LMFunctionConcurrency`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-LMFunctionConcurrency`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Function Concurrency der Lambda-Funktion entfernt.

```
Remove-LMFunctionConcurrency -FunctionName "MylambdaFunction123"
```

- Einzelheiten zur API finden Sie unter DeleteFunctionConcurrency AWS Tools for PowerShellCmdlet-Referenz.

`Remove-LMPermission`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-LMPermission`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Funktionsrichtlinie für die angegebene StatementId Lambda-Funktion entfernt.

```
$policy =  Get-LMPolicy -FunctionName "MylambdaFunction123" -Select Policy |
 ConvertFrom-Json| Select-Object -ExpandProperty Statement
Remove-LMPermission -FunctionName "MylambdaFunction123" -StatementId $policy[0].Sid
```

- Einzelheiten zur API finden Sie unter RemovePermission AWS Tools for PowerShellCmdlet-Referenz.

`Remove-LMProvisionedConcurrencyConfig`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-LMProvisionedConcurrencyConfig`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Provisioned Concurrency Configuration für einen bestimmten Alias entfernt.

```
Remove-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -Qualifier
 "NewAlias1"
```

- Einzelheiten zur API finden Sie unter DeleteProvisionedConcurrencyConfig AWS Tools for PowerShellCmdlet-Referenz.

`Remove-LMResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-LMResourceTag`

Tools für PowerShell

Beispiel 1: Entfernt die bereitgestellten Tags aus einer Funktion. Das Cmdlet fordert Sie zur Bestätigung auf, bevor der Vorgang fortgesetzt wird, sofern der Schalter -Force nicht angegeben ist. Es wird ein einziger Aufruf an den Dienst gesendet, um die Tags zu entfernen.

```
Remove-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction" -TagKey "Washington","Oregon","California"
```

Beispiel 2: Entfernt die bereitgestellten Tags aus einer Funktion. Das Cmdlet fordert Sie zur Bestätigung auf, bevor der Vorgang fortgesetzt wird, sofern der Schalter -Force nicht angegeben ist. Sobald der Dienst pro bereitgestelltem Tag aufgerufen wurde.

```
"Washington","Oregon","California" | Remove-LMResourceTag -Resource
 "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
```

- Einzelheiten zur API finden Sie unter UntagResource AWS Tools for PowerShellCmdlet-Referenz.

`Update-LMAlias`

Das folgende Codebeispiel zeigt die Verwendung. `Update-LMAlias`

Tools für PowerShell

Beispiel 1: Dieses Beispiel aktualisiert die Konfiguration einer vorhandenen Lambda-Funktion Alias. Der RoutingConfiguration Wert wird aktualisiert, sodass 60% (0,6) des Datenverkehrs auf Version 1 umgestellt werden

```
Update-LMAlias -FunctionName "MylambdaFunction123" -Description " Alias for version
 2" -FunctionVersion 2 -Name "newlabel1" -RoutingConfig_AdditionalVersionWeight
 @{Name="1";Value="0.6}
```

- Einzelheiten zur API finden Sie unter UpdateAlias AWS Tools for PowerShellCmdlet-Referenz.

`Update-LMFunctionCode`

Das folgende Codebeispiel zeigt die Verwendung. `Update-LMFunctionCode`

Tools für PowerShell

Beispiel 1: Aktualisiert die Funktion mit dem Namen 'MyFunction' mit neuem Inhalt, der in der angegebenen ZIP-Datei enthalten ist. Für eine C#.NET Core Lambda-Funktion sollte die ZIP-Datei die kompilierte Assembly enthalten.

```
Update-LMFunctionCode -FunctionName MyFunction -ZipFilename .\UpdatedCode.zip
```

Beispiel 2: Dieses Beispiel ähnelt dem vorherigen, verwendet jedoch ein Amazon S3 S3-Objekt, das den aktualisierten Code enthält, um die Funktion zu aktualisieren.

```
Update-LMFunctionCode -FunctionName MyFunction -BucketName mybucket -Key
 UpdatedCode.zip
```

- Einzelheiten zur API finden Sie unter UpdateFunctionCode AWS Tools for PowerShellCmdlet-Referenz.

`Update-LMFunctionConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Update-LMFunctionConfiguration`

Tools für PowerShell

Beispiel 1: Dieses Beispiel aktualisiert die bestehende Lambda-Funktionskonfiguration

```
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
```

- Einzelheiten zur API finden Sie unter UpdateFunctionConfiguration AWS Tools for PowerShellCmdlet-Referenz.

`Write-LMFunctionConcurrency`

Das folgende Codebeispiel zeigt die Verwendung. `Write-LMFunctionConcurrency`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Parallelitätseinstellungen für die gesamte Funktion angewendet.

```
Write-LMFunctionConcurrency -FunctionName "MylambdaFunction123" -
ReservedConcurrentExecution 100
```

- Einzelheiten zur API finden Sie unter PutFunctionConcurrency AWS Tools for PowerShellCmdlet-Referenz.

`Write-LMProvisionedConcurrencyConfig`

Das folgende Codebeispiel zeigt die Verwendung. `Write-LMProvisionedConcurrencyConfig`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird dem Alias einer Funktion eine bereitgestellte Parallelitätskonfiguration hinzugefügt

```
Write-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -
ProvisionedConcurrentExecution 20 -Qualifier "NewAlias1"
```

- Einzelheiten zur API finden Sie unter PutProvisionedConcurrencyConfig AWS Tools for PowerShellCmdlet-Referenz.

Amazon ML-Beispiele mit Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon ML Aktionen ausführen und allgemeine Szenarien implementieren. AWS Tools for PowerShell

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`Get-MLBatchPrediction`

Das folgende Codebeispiel zeigt die Verwendung`Get-MLBatchPrediction`.

Tools für PowerShell

Beispiel 1: Gibt die detaillierten Metadaten für eine Batch-Vorhersage mit der ID-ID zurück.

```
Get-MLBatchPrediction -BatchPredictionId ID
```

- Einzelheiten zur API finden Sie unter GetBatchPrediction AWS Tools for PowerShellCmdlet-Referenz.

`Get-MLBatchPredictionList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-MLBatchPredictionList`

Tools für PowerShell

Beispiel 1: Gibt eine Liste aller BatchPredictions und der zugehörigen Datensätze zurück, die dem in der Anfrage angegebenen Suchkriterium entsprechen.

```
Get-MLBatchPredictionList
```

Beispiel 2: Gibt eine Liste aller BatchPredictions mit dem Status ABGESCHLOSSEN zurück.

```
Get-MLBatchPredictionList -FilterVariable Status -EQ COMPLETED
```

- Einzelheiten zur API finden Sie unter DescribeBatchPredictions AWS Tools for PowerShellCmdlet-Referenz.

`Get-MLDataSource`

Das folgende Codebeispiel zeigt die Verwendung. `Get-MLDataSource`

Tools für PowerShell

Beispiel 1: Gibt die Metadaten, den Status und die Datendateiinformationen für a DataSource mit der ID-ID zurück

```
Get-MLDataSource -DataSourceId ID
```

- Einzelheiten zur API finden Sie unter GetDataSource AWS Tools for PowerShellCmdlet-Referenz.

`Get-MLDataSourceList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-MLDataSourceList`

Tools für PowerShell

Beispiel 1: Gibt eine Liste aller DataSources und der zugehörigen Datensätze zurück.

```
Get-MLDataSourceList
```

Beispiel 2: Gibt eine Liste aller DataSources mit dem Status ABGESCHLOSSEN zurück.

```
Get-MLDataDourceList -FilterVariable Status -EQ COMPLETED
```

- Einzelheiten zur API finden Sie unter DescribeDataSources AWS Tools for PowerShellCmdlet-Referenz.

`Get-MLEvaluation`

Das folgende Codebeispiel zeigt die Verwendung. `Get-MLEvaluation`

Tools für PowerShell

Beispiel 1: Gibt Metadaten und Status für eine Bewertung mit ID-ID zurück.

```
Get-MLEvaluation -EvaluationId ID
```

- Einzelheiten zur API finden Sie unter GetEvaluation AWS Tools for PowerShellCmdlet-Referenz.

`Get-MLEvaluationList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-MLEvaluationList`

Tools für PowerShell

Beispiel 1: Gibt eine Liste aller Evaluierungsressourcen zurück

```
Get-MLEvaluationList
```

Beispiel 2: Gibt eine Liste aller Evaluationen mit dem Status ABGESCHLOSSEN zurück.

```
Get-MLEvaluationList -FilterVariable Status -EQ COMPLETED
```

- Einzelheiten zur API finden Sie unter DescribeEvaluations AWS Tools for PowerShellCmdlet-Referenz.

`Get-MLModel`

Das folgende Codebeispiel zeigt die Verwendung. `Get-MLModel`

Tools für PowerShell

Beispiel 1: Gibt die detaillierten Metadaten, den Status, das Schema und die Datendateiinformationen für ein MLModel mit der ID-ID zurück.

```
Get-MLModel -ModelId ID
```

- API-Details finden Sie unter GetMLModel in der AWS Tools for PowerShell Cmdlet-Referenz.

`Get-MLModelList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-MLModelList`

Tools für PowerShell

Beispiel 1: Gibt eine Liste aller Modelle und der zugehörigen Datensätze zurück.

```
Get-MLModelList
```

Beispiel 2: Gibt eine Liste aller Modelle mit dem Status ABGESCHLOSSEN zurück.

```
Get-MLModelList -FilterVariable Status -EQ COMPLETED
```

- Einzelheiten zur API finden Sie unter DescribeMLModels in AWS Tools for PowerShell der Cmdlet-Referenz.

`Get-MLPrediction`

Das folgende Codebeispiel zeigt die Verwendung. `Get-MLPrediction`

Tools für PowerShell

Beispiel 1: Senden Sie einen Datensatz an die URL des Echtzeit-Prognoseendpunkts für das Modell mit der ID-ID.

```
Get-MLPrediction -ModelId ID -PredictEndpoint URL -Record @{"A" = "B"; "C" = "D";}
```

- Einzelheiten zur API finden Sie unter Predict in der AWS Tools for PowerShell Cmdlet-Referenz.

`New-MLBatchPrediction`

Das folgende Codebeispiel zeigt die Verwendung. `New-MLBatchPrediction`

Tools für PowerShell

Beispiel 1: Erstellen Sie eine neue Batch-Prognoseanforderung für ein Modell mit der ID-ID und platzieren Sie die Ausgabe am angegebenen S3-Speicherort.

```
New-MLBatchPrediction -ModelId ID -Name NAME -OutputURI s3://...
```

- Einzelheiten zur API finden Sie unter CreateBatchPrediction AWS Tools for PowerShellCmdlet-Referenz.

`New-MLDataSourceFromS3`

Das folgende Codebeispiel zeigt die Verwendung. `New-MLDataSourceFromS3`

Tools für PowerShell

Beispiel 1: Erstellen Sie eine Datenquelle mit Daten für einen S3-Standort mit dem Namen NAME und dem Schema SCHEMA.

```
New-MLDataSourceFromS3 -Name NAME -ComputeStatistics $true -DataSpec_DataLocationS3
 "s3://BUCKET/KEY" -DataSchema SCHEMA
```

- API-Details finden Sie unter CreateDataSourceFromS3 in der AWS Tools for PowerShell Cmdlet-Referenz.

`New-MLEvaluation`

Das folgende Codebeispiel zeigt die Verwendung. `New-MLEvaluation`

Tools für PowerShell

Beispiel 1: Erstellen Sie eine Auswertung für eine bestimmte Datenquellen-ID und Modell-ID

```
New-MLEvaluation -Name NAME -DataSourceId DSID -ModelId MID
```

- Einzelheiten zur API finden Sie unter CreateEvaluation AWS Tools for PowerShellCmdlet-Referenz.

`New-MLModel`

Das folgende Codebeispiel zeigt die Verwendung. `New-MLModel`

Tools für PowerShell

Beispiel 1: Erstellen Sie ein neues Modell mit Trainingsdaten.

```
New-MLModel -Name NAME -ModelType BINARY -Parameter @{...} -TrainingDataSourceId ID
```

- API-Details finden Sie unter CreateMLModel in AWS Tools for PowerShell der Cmdlet-Referenz.

`New-MLRealtimeEndpoint`

Das folgende Codebeispiel zeigt die Verwendung. `New-MLRealtimeEndpoint`

Tools für PowerShell

Beispiel 1: Erstellen Sie einen neuen Endpunkt für Echtzeitprognosen für die angegebene Modell-ID.

```
New-MLRealtimeEndpoint -ModelId ID
```

- Einzelheiten zur API finden Sie unter CreateRealtimeEndpoint AWS Tools for PowerShellCmdlet-Referenz.

Macie-Beispiele mit Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS Tools for PowerShell mit Macie Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

Aktionen

`Get-MAC2FindingList`

Das folgende Codebeispiel zeigt die Verwendung`Get-MAC2FindingList`.

Tools für PowerShell

Beispiel 1: Gibt eine Liste mit Ergebnissen zurück, die FindingIds eine Erkennung sensibler Daten mit dem Typ „CREDIT_CARD_NUMBER“ oder „US_SOCIAL_SECURITY_NUMBER“ enthalten

```
$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
```

- Einzelheiten ListFindings AWS Tools for PowerShellzur API finden Sie unter Cmdlet-Referenz.

AWS OpsWorks Beispiele für die Verwendung von Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS Tools for PowerShell with Aktionen ausführen und allgemeine Szenarien implementieren AWS OpsWorks.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`New-OPSDeployment`

Das folgende Codebeispiel zeigt die Verwendung`New-OPSDeployment`.

Tools für PowerShell

Beispiel 1: Dieser Befehl erstellt eine neue App-Bereitstellung auf allen Linux-basierten Instanzen in einer Ebene in AWS OpsWorks Stacks. Auch wenn Sie eine Layer-ID angeben, müssen Sie auch eine Stack-ID angeben. Mit dem Befehl kann die Bereitstellung die Instanzen bei Bedarf neu starten.

```
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"}
```

Beispiel 2: Dieser Befehl stellt das **appsetup** Rezept aus dem **phpapp** Kochbuch und das **secbaseline** Rezept aus dem **testcookbook** Kochbuch bereit. Das Bereitstellungsziel ist eine Instanz, aber die Stack-ID und die Layer-ID sind ebenfalls erforderlich. Das **allow_reboot** Parameterattribut Command_Arg ist auf gesetzt**true**, sodass die Bereitstellung die Instanzen bei Bedarf neu starten kann.

```
$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
```

- Einzelheiten zur API finden Sie unter CreateDeploymentCmdlet-Referenz.AWS Tools for PowerShell

AWS-Preisliste Beispiele für die Verwendung von Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS Tools for PowerShell with Aktionen ausführen und allgemeine Szenarien implementieren AWS-Preisliste.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`Get-PLSAttributeValue`

Das folgende Codebeispiel zeigt die Verwendung`Get-PLSAttributeValue`.

Tools für PowerShell

Beispiel 1: Gibt die Werte für das Attribut 'volumeType' für Amazon EC2 in der Region us-east-1 zurück.

```
Get-PLSAttributeValue -ServiceCode AmazonEC2 -AttributeName "volumeType" -region us-
east-1
```

Ausgabe:

```
Value
-----
Cold HDD
General Purpose
Magnetic
Provisioned IOPS
Throughput Optimized HDD
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz. GetAttributeValuesAWS Tools for PowerShell

`Get-PLSProduct`

Das folgende Codebeispiel zeigt die Verwendung. `Get-PLSProduct`

Tools für PowerShell

Beispiel 1: Gibt Details aller Produkte für Amazon EC2 zurück.

```
Get-PLSProduct -ServiceCode AmazonEC2 -Region us-east-1
```

Ausgabe:

```
{"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"},...
```

Beispiel 2: Gibt Daten für Amazon EC2 in der Region us-east-1 zurück, gefiltert nach Volumetypen von „General Purpose“, die SSD-gestützt sind.

```
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
```

Ausgabe:

```
{"product":{"productFamily":"Storage","attributes":{"storageMedia":"SSD-
backed","maxThroughputvolume":"160 MB/sec","volumeType":"General
 Purpose","maxIopsvolume":"10000",...
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz. GetProductsAWS Tools for PowerShell

`Get-PLSService`

Das folgende Codebeispiel zeigt die Verwendung. `Get-PLSService`

Tools für PowerShell

Beispiel 1: Gibt die Metadaten für alle verfügbaren Servicecodes in der Region us-east-1 zurück.

```
Get-PLSService -Region us-east-1
```

Ausgabe:

```
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
...
```

Beispiel 2: Gibt die Metadaten für den Amazon EC2-Service in der Region us-east-1 zurück.

```
Get-PLSService -ServiceCode AmazonEC2 -Region us-east-1
```

Ausgabe:

```
AttributeNames                                                      ServiceCode
--------------                                                      -----------
{volumeType, maxIopsvolume, instanceCapacity10xlarge, locationType...} AmazonEC2
```

- Einzelheiten zur API finden Sie unter DescribeServicesCmdlet-Referenz.AWS Tools for PowerShell

Beispiele für Resource Groups mit Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS Tools for PowerShell with Resource Groups Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`Add-RGResourceTag`

Das folgende Codebeispiel zeigt die Verwendung`Add-RGResourceTag`.

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der angegebenen Ressourcengruppe arn der Tag-Schlüssel 'Instances' mit dem Wert 'workboxes' hinzugefügt

```
Add-RGResourceTag -Tag @{Instances="workboxes"} -Arn arn:aws:resource-groups:eu-
west-1:123456789012:group/workboxes
```

Ausgabe:

```
Arn                                                                 Tags
---                                                                 ----
arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes {[Instances,
 workboxes]}
```

- Einzelheiten zur API finden Sie unter Tag in der AWS Tools for PowerShell Cmdlet-Referenz.

`Find-RGResource`

Das folgende Codebeispiel zeigt die Verwendung. `Find-RGResource`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Ressourcentyp ResourceQuery für Instanz mit Tag-Filtern erstellt und Ressourcen gesucht.

```
$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
```

Ausgabe:

```
ResourceArn                                                     ResourceType
-----------                                                     ------------
arn:aws:ec2:eu-west-1:123456789012:instance/i-0123445b6cb7bd67b AWS::EC2::Instance
```

- Einzelheiten zur API finden Sie unter SearchResources AWS Tools for PowerShellCmdlet-Referenz.

`Get-RGGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Get-RGGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Ressourcengruppe gemäß dem Gruppennamen abgerufen

```
Get-RGGroup -GroupName auto-no
```

Ausgabe:

```
Description GroupArn                                                Name
----------- --------                                                ----
            arn:aws:resource-groups:eu-west-1:123456789012:group/auto-no auto-no
```

- Einzelheiten zur API finden Sie unter GetGroup AWS Tools for PowerShellCmdlet-Referenz.

`Get-RGGroupList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-RGGroupList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Ressourcengruppe aufgeführt, die bereits erstellt wurde.

```
Get-RGGroupList
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter ListGroups AWS Tools for PowerShellCmdlet-Referenz.

`Get-RGGroupQuery`

Das folgende Codebeispiel zeigt die Verwendung. `Get-RGGroupQuery`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Ressourcenabfrage für die angegebene Ressourcengruppe abgerufen

```
Get-RGGroupQuery -GroupName auto-no | Select-Object -ExpandProperty ResourceQuery
```

Ausgabe:

```
Query
         Type
-----
         ----
{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":[{"Key":"auto","Values":
["no"]}]} TAG_FILTERS_1_0
```

- Einzelheiten zur API finden Sie unter GetGroupQuery AWS Tools for PowerShellCmdlet-Referenz.

`Get-RGGroupResourceList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-RGGroupResourceList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Gruppenressourcen nach Ressourcentyp gefiltert aufgelistet

```
Get-RGGroupResourceList -Filter @{Name="resource-type";Values="AWS::EC2::Instance"}
 -GroupName auto-yes | Select-Object -ExpandProperty ResourceIdentifiers
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter ListGroupResources AWS Tools for PowerShellCmdlet-Referenz.

`Get-RGResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Get-RGResourceTag`

Tools für PowerShell

Beispiel 1: Dieses Beispiel listet Tags für die angegebene Ressourcengruppe arn auf

```
Get-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/
workboxes
```

Ausgabe:

```
Key       Value
---       -----
Instances workboxes
```

- Einzelheiten zur API finden Sie unter GetTags AWS Tools for PowerShellCmdlet-Referenz.

`New-RGGroup`

Das folgende Codebeispiel zeigt die Verwendung. `New-RGGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine neue tagbasierte AWS Ressourcengruppe für Resource Groups mit dem Namen TestPowerShellGroup erstellt. Die Gruppe umfasst Amazon EC2 EC2-Instances in der aktuellen Region, die mit dem Tag-Schlüssel „Name“ und dem Tag-Wert „test2"

gekennzeichnet sind. Der Befehl gibt die Abfrage und den Typ der Gruppe sowie die Ergebnisse des Vorgangs zurück.

```
$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."
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter CreateGroup AWS Tools for PowerShellCmdlet-Referenz.

`Remove-RGGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-RGGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die benannte Ressourcengruppe entfernt

```
Remove-RGGroup -GroupName non-tag-cfn-elbv2
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DeleteGroup AWS Tools for PowerShellCmdlet-Referenz.

`Remove-RGResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-RGResourceTag`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das erwähnte Tag aus der Ressourcengruppe entfernt

```
Remove-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/
workboxes -Key Instances
```

Ausgabe:

```
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}
```

- Einzelheiten zur API finden Sie unter Untag in der AWS Tools for PowerShell Cmdlet-Referenz.

`Update-RGGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Update-RGGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Beschreibung der Gruppe aktualisiert

```
Update-RGGroup -GroupName auto-yes -Description "Instances auto-remove"
```

Ausgabe:

```
Description             GroupArn
  Name
-----------             --------
  ----
Instances to be cleaned arn:aws:resource-groups:eu-west-1:123456789012:group/auto-
yes auto-yes
```

- Einzelheiten zur API finden Sie unter UpdateGroup AWS Tools for PowerShellCmdlet-Referenz.

`Update-RGGroupQuery`

Das folgende Codebeispiel zeigt die Verwendung. `Update-RGGroupQuery`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Abfrageobjekt erstellt und die Abfrage für die Gruppe aktualisiert.

```
$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
```

Ausgabe:

```
GroupName ResourceQuery
--------- -------------
build600  Amazon.ResourceGroups.Model.ResourceQuery
```

- Einzelheiten zur API finden Sie unter UpdateGroupQuery AWS Tools for PowerShellCmdlet-Referenz.

API-Beispiele für das Tagging von Resource Groups mithilfe von Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe der AWS Tools for PowerShell with Resource Groups Tagging API Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, über den Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`Add-RGTResourceTag`

Das folgende Codebeispiel zeigt die Verwendung`Add-RGTResourceTag`.

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Tag-Schlüssel „stage“ und „version“ mit den Werten „beta“ und „preprod_test“ zu einem Amazon S3 S3-Bucket und einer Amazon DynamoDB-Tabelle hinzugefügt. Ein einziger Aufruf erfolgt an den Service, um die Tags anzuwenden.

```
$arn1 = "arn:aws:s3:::mybucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

Add-RGTResourceTag -ResourceARNList $arn1,$arn2 -Tag @{ "stage"="beta";
 "version"="preprod_test" }
```

Beispiel 2: Dieses Beispiel fügt die angegebenen Tags und Werte zu einem Amazon S3 S3-Bucket und einer Amazon DynamoDB-Tabelle hinzu. Der Dienst wird zweimal aufgerufen, einer für jeden Ressourcen-ARN, der über die Pipeline an das Cmdlet übergeben wird.

```
$arn1 = "arn:aws:s3:::mybucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

$arn1,$arn2 | Add-RGTResourceTag -Tag @{ "stage"="beta"; "version"="preprod_test" }
```

- Einzelheiten zur API finden Sie unter TagResourcesCmdlet-Referenz.AWS Tools for PowerShell

`Get-RGTResource`

Das folgende Codebeispiel zeigt die Verwendung. `Get-RGTResource`

Tools für PowerShell

Beispiel 1: Gibt alle markierten Ressourcen in einer Region und die mit der Ressource verknüpften Tag-Schlüssel zurück. Wenn dem Cmdlet kein -Region-Parameter angegeben wird, versucht es, die Region aus den Metadaten der Shell oder der EC2-Instanz abzuleiten.

```
Get-RGTResource
```

Ausgabe:

```
ResourceARN                                                    Tags
-----------                                                    ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable          {stage, version}
arn:aws:s3:::mybucket                                          {stage, version,
 othertag}
```

Beispiel 2: Gibt alle markierten Ressourcen des angegebenen Typs in einer Region zurück. Die Zeichenfolge für jeden Servicenamen und Ressourcentyp entspricht der Zeichenfolge, die im Amazon-Ressourcennamen (ARN) einer Ressource eingebettet ist.

```
Get-RGTResource -ResourceType "s3"
```

Ausgabe:

```
ResourceARN                                                    Tags
-----------                                                    ----
arn:aws:s3:::mybucket                                          {stage, version,
 othertag}
```

Beispiel 3: Gibt alle markierten Ressourcen des angegebenen Typs in einer Region zurück. Beachten Sie, dass, wenn die Ressourcentypen über die Pipeline an das Cmdlet übergeben werden, für jeden angegebenen Ressourcentyp ein Aufruf an den Dienst erfolgt.

```
"dynamodb","s3" | Get-RGTResource
```

Ausgabe:

```
ResourceARN                                                    Tags
-----------                                                    ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable          {stage, version}
arn:aws:s3:::mybucket                                          {stage, version,
 othertag}
```

Beispiel 4: Gibt alle markierten Ressourcen zurück, die dem angegebenen Filter entsprechen.

```
Get-RGTResource -TagFilter @{ Key="stage" }
```

Ausgabe:

```
ResourceARN                                                    Tags
-----------                                                    ----
arn:aws:s3:::mybucket                                          {stage, version,
 othertag}
```

Beispiel 5: Gibt alle markierten Ressourcen zurück, die dem angegebenen Filter und Ressourcentyp entsprechen.

```
Get-RGTResource -TagFilter @{ Key="stage" } -ResourceType "dynamodb"
```

Ausgabe:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable           {stage, version}
```

Beispiel 6: Gibt alle markierten Ressourcen zurück, die dem angegebenen Filter entsprechen.

```
Get-RGTResource -TagFilter @{ Key="stage"; Values=@("beta","gamma") }
```

Ausgabe:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable           {stage, version}
```

- Einzelheiten zur API finden Sie unter GetResources AWS Tools for PowerShellCmdlet-Referenz.

`Get-RGTTagKey`

Das folgende Codebeispiel zeigt die Verwendung. `Get-RGTTagKey`

Tools für PowerShell

Beispiel 1: Gibt alle Tag-Schlüssel in der angegebenen Region zurück. Wenn der Parameter -Region nicht angegeben ist, versucht das Cmdlet, die Region aus der Standard-Shellregion oder den Metadaten der EC2-Instanz abzuleiten. Beachten Sie, dass die Tag-Schlüssel nicht in einer bestimmten Reihenfolge zurückgegeben werden.

```
Get-RGTTagKey -region us-west-2
```

Ausgabe:

```
version
stage
```

- Einzelheiten zur API finden Sie unter GetTagKeys AWS Tools for PowerShellCmdlet-Referenz.

`Get-RGTTagValue`

Das folgende Codebeispiel zeigt die Verwendung. `Get-RGTTagValue`

Tools für PowerShell

Beispiel 1: Gibt den Wert für das angegebene Tag in einer Region zurück. Wenn der Parameter -Region nicht angegeben ist, versucht das Cmdlet, die Region aus der Standard-Shellregion oder den Metadaten der EC2-Instanz abzuleiten.

```
Get-RGTTagValue -Key "stage" -Region us-west-2
```

Ausgabe:

```
beta
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz. GetTagValuesAWS Tools for PowerShell

`Remove-RGTResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-RGTResourceTag`

Tools für PowerShell

Beispiel 1: Entfernt die Tag-Schlüssel „Stage“ und „Version“ sowie die zugehörigen Werte aus einem Amazon S3 S3-Bucket und einer Amazon DynamoDB-Tabelle. Ein einziger Aufruf erfolgt an den Service, um die Tags zu entfernen. Bevor die Tags entfernt werden, fordert das Cmdlet zur Bestätigung auf. Um die Bestätigung zu umgehen, fügen Sie den Parameter -Force hinzu.

```
$arn1 = "arn:aws:s3:::mybucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

Remove-RGTResourceTag -ResourceARNList $arn1,$arn2 -TagKey "stage","version"
```

Beispiel 2: Entfernt die Tag-Schlüssel „Stage“ und „Version“ sowie die zugehörigen Werte aus einem Amazon S3 S3-Bucket und einer Amazon DynamoDB-Tabelle. Der Dienst wird zweimal

aufgerufen, einer für jeden Ressourcen-ARN, der über die Pipeline an das Cmdlet übergeben wird. Vor jedem Aufruf fordert das Cmdlet zur Bestätigung auf. Um die Bestätigung zu umgehen, fügen Sie den Parameter -Force hinzu.

```
$arn1 = "arn:aws:s3:::mybucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

$arn1,$arn2 | Remove-RGTResourceTag -TagKey "stage","version"
```

- Einzelheiten zur API finden Sie unter UntagResources AWS Tools for PowerShellCmdlet-Referenz.

Route 53-Beispiele mit Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Route 53 Aktionen ausführen und allgemeine Szenarien implementieren. AWS Tools for PowerShell

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`Edit-R53ResourceRecordSet`

Das folgende Codebeispiel zeigt die Verwendung`Edit-R53ResourceRecordSet`.

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein A-Record für www.example.com erstellt und der A-Record für test.example.com von 192.0.2.3 auf 192.0.2.1 geändert. Beachten Sie, dass Werte für

Datensätze vom Typ TXT in doppelten Anführungszeichen stehen müssen. Weitere Informationen finden Sie in der Dokumentation zu Amazon Route 53 Sie können das Get-R53Change Cmdlet verwenden, um abzufragen, wann die Änderungen abgeschlossen sind.

```
$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
```

Beispiel 2: Dieses Beispiel zeigt, wie Alias-Ressourcendatensätze erstellt werden. 'Z222222222' ist die ID der von Amazon Route 53 gehosteten Zone, in der Sie den Alias-Ressourcendatensatz erstellen. 'example.com' ist der Zonen-Apex, für den Sie einen Alias erstellen möchten, und 'www.example.com' ist eine Subdomain, für die Sie auch einen Alias erstellen möchten. 'Z1111111111111' ist ein Beispiel für eine Hosting-Zonen-ID für den Load Balancer und

'example-load-balancer-1111111111.us-east-1.elb.amazonaws.com' ist ein Beispiel für einen Load Balancer-Domainnamen, mit dem Amazon Route 53 auf Anfragen für example.com und www.example.com antwortet. Weitere Informationen finden Sie in der Dokumentation zu Amazon Route 53 Sie können das Get-R53Change Cmdlet verwenden, um abzufragen, wann die Änderungen abgeschlossen sind.

```
$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
```

Beispiel 3: In diesem Beispiel werden zwei A-Datensätze für www.example.com erstellt. In einem Viertel der Fälle (1/ (1+3)) beantwortet Amazon Route 53 Anfragen für www.example.com mit den beiden Werten für den ersten Ressourcendatensatz (192.0.2.9 und 192.0.2.10). In drei Vierteln der Fälle (3/ (1+3)) beantwortet Amazon Route 53 Anfragen für www.example.com mit den beiden

Werten für den zweiten Ressourcendatensatz (192.0.2.11 und 192.0.2.12). Weitere Informationen finden Sie in der Dokumentation zu Amazon Route 53 Sie können das Get-R53Change Cmdlet verwenden, um abzufragen, wann die Änderungen abgeschlossen sind.

```
$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
```

Beispiel 4: Dieses Beispiel zeigt, wie gewichtete Alias-Ressourcendatensätze erstellt werden, vorausgesetzt, dass example.com die Domain ist, für die Sie gewichtete Alias-Ressourcendatensätze erstellen möchten. SetIdentifier unterscheidet die beiden gewichteten Alias-Ressourcendatensätze voneinander. Dieses Element ist erforderlich, da die Elemente Name und Type für beide Ressourcendatensätze dieselben Werte haben. Z1111111111111 und Z3333333333333 sind Beispiele für Hosting-Zonen-IDs für den ELB-Load Balancer, die durch den Wert von dnsName angegeben werden. example-

load-balancer-2222222222.us-east-1.elb.amazonaws.com und example-load-balancer -4444444444.us-east-1.elb.amazonaws.com sind Beispiele für Elastic Load Balancing Balancing-Domains, von denen Amazon Route 53 auf Anfragen für example.com antwortet. Weitere Informationen finden Sie in der Dokumentation zu Amazon Route 53 Sie können das Get-R53Change Cmdlet verwenden, um abzufragen, wann die Änderungen abgeschlossen sind.

```
$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
```

Beispiel 5: In diesem Beispiel werden zwei Latenz-Alias-Ressourcendatensätze erstellt, einen für einen ELB-Load Balancer in der Region USA West (Oregon) (us-west-2) und einen weiteren für einen Load Balancer in der Region Asien-Pazifik (Singapur) (ap-southeast-1). Weitere Informationen finden Sie in der Dokumentation zu Amazon Route 53 Sie können das Get-R53Change Cmdlet verwenden, um abzufragen, wann die Änderungen abgeschlossen sind.

```
$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
```

- Einzelheiten zur API finden Sie unter ChangeResourceRecordSets AWS Tools for PowerShellCmdlet-Referenz.

`Get-R53AccountLimit`

Das folgende Codebeispiel zeigt die Verwendung. `Get-R53AccountLimit`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die maximale Anzahl von Hosting-Zonen zurückgegeben, die mit dem aktuellen Konto erstellt werden können.

```
Get-R53AccountLimit -Type MAX_HOSTED_ZONES_BY_OWNER
```

Ausgabe:

```
15
```

- Einzelheiten zur API finden Sie unter GetAccountLimit AWS Tools for PowerShellCmdlet-Referenz.

`Get-R53CheckerIpRanges`

Das folgende Codebeispiel zeigt die Verwendung. `Get-R53CheckerIpRanges`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die CIDRs für die Route53-Zustandsprüfungen zurückgegeben

```
Get-R53CheckerIpRanges
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter GetCheckerIpRangesCmdlet-Referenz.AWS Tools for PowerShell

`Get-R53HostedZone`

Das folgende Codebeispiel zeigt die Verwendung. `Get-R53HostedZone`

Tools für PowerShell

Beispiel 1: Gibt Details der Hosting-Zone mit der ID Z1D633PJN98FT9 zurück.

```
Get-R53HostedZone -Id Z1D633PJN98FT9
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz. GetHostedZoneAWS Tools for PowerShell

`Get-R53HostedZoneCount`

Das folgende Codebeispiel zeigt die Verwendung. `Get-R53HostedZoneCount`

Tools für PowerShell

Beispiel 1: Gibt die Gesamtzahl der öffentlichen und privaten gehosteten Zonen für die aktuelle Version zurück AWS-Konto.

```
Get-R53HostedZoneCount
```

- Einzelheiten zur API finden Sie unter GetHostedZoneCount AWS Tools for PowerShellCmdlet-Referenz.

`Get-R53HostedZoneLimit`

Das folgende Codebeispiel zeigt die Verwendung. `Get-R53HostedZoneLimit`

Tools für PowerShell

Beispiel 1: Dieses Beispiel gibt die Obergrenze für die maximale Anzahl von Datensätzen zurück, die in der angegebenen Hosting-Zone erstellt werden können.

```
Get-R53HostedZoneLimit -HostedZoneId Z3MEQ8T7HAAAAF -Type MAX_RRSETS_BY_ZONE
```

Ausgabe:

```
5
```

- Einzelheiten zur API finden Sie unter GetHostedZoneLimit AWS Tools for PowerShellCmdlet-Referenz.

`Get-R53HostedZoneList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-R53HostedZoneList`

Tools für PowerShell

Beispiel 1: Gibt alle Ihre öffentlichen und privaten Hosting-Zonen aus.

```
Get-R53HostedZoneList
```

Beispiel 2: Gibt alle gehosteten Zonen aus, die dem wiederverwendbaren Delegierungssatz mit der ID NZ8X2CISAMPLE zugeordnet sind

```
Get-R53HostedZoneList -DelegationSetId NZ8X2CISAMPLE
```

- Einzelheiten zur API finden Sie unter ListHostedZonesCmdlet-Referenz.AWS Tools for PowerShell

`Get-R53HostedZonesByName`

Das folgende Codebeispiel zeigt die Verwendung. `Get-R53HostedZonesByName`

Tools für PowerShell

Beispiel 1: Gibt alle Ihre öffentlichen und privaten Hosting-Zonen in ASCII-Reihenfolge nach Domainnamen zurück.

```
Get-R53HostedZonesByName
```

Beispiel 2: Gibt Ihre öffentlichen und privaten gehosteten Zonen in ASCII-Reihenfolge nach Domainnamen zurück, beginnend mit dem angegebenen DNS-Namen.

```
Get-R53HostedZonesByName -DnsName example2.com
```

Beispiel 3: Dieses Beispiel zeigt, wie Sie die gehosteten Zonen manuell auflisten, indem Sie zuerst ein einzelnes Element abrufen und dann zwei nacheinander iterieren, bis alle Zonen zurückgegeben wurden. Dabei werden Markereigenschaften verwendet, die nach jedem Aufruf an die Serviceantwort im Stack angehängt wurden. **`$AWSHistory`**

```
Get-R53HostedZonesByName -MaxItem 1
while ($LastServiceResponse.IsTruncated)
{
    $nextPageParams = @{
        DnsName=$LastServiceResponse.NextDNSName
        HostedZoneId=$LastServiceResponse.NextHostedZoneId
    }
    Get-R53HostedZonesByName -MaxItem 2 @nextPageParams
```

```
}
```

- Einzelheiten zur API finden Sie unter ListHostedZonesByNameCmdlet-Referenz.AWS Tools for PowerShell

`Get-R53QueryLoggingConfigList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-R53QueryLoggingConfigList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden alle Konfigurationen für die DNS-Abfrageprotokollierung zurückgegeben, die der aktuellen Version zugeordnet sind AWS-Konto.

```
Get-R53QueryLoggingConfigList
```

Ausgabe:

```
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:*
```

- Einzelheiten zur API finden Sie unter ListQueryLoggingConfigs AWS Tools for PowerShellCmdlet-Referenz.

`Get-R53ReusableDelegationSet`

Das folgende Codebeispiel zeigt die Verwendung. `Get-R53ReusableDelegationSet`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Informationen über den angegebenen Delegierungssatz abgerufen, einschließlich der vier Namenserver, die dem Delegierungssatz zugewiesen sind.

```
Get-R53ReusableDelegationSet -Id N23DS9X4AYEAAA
```

Ausgabe:

```
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}
```

- Einzelheiten zur API finden Sie unter GetReusableDelegationSet AWS Tools for PowerShellCmdlet-Referenz.

`New-R53HostedZone`

Das folgende Codebeispiel zeigt die Verwendung. `New-R53HostedZone`

Tools für PowerShell

Beispiel 1: Erstellt eine neue gehostete Zone mit dem Namen 'example.com', die einem wiederverwendbaren Delegierungssatz zugeordnet ist. Beachten Sie, dass Sie einen Wert für den CallerReference Parameter angeben müssen, damit Anfragen, die erforderlich sind, bei Bedarf erneut versucht werden müssen, ohne dass das Risiko besteht, dass der Vorgang zweimal ausgeführt wird. Da die gehostete Zone in einer VPC erstellt wird, ist sie automatisch privat und Sie sollten den PrivateZone Parameter - HostedZoneConfig _ nicht festlegen.

```
$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
```

- Einzelheiten zur API finden Sie unter CreateHostedZone AWS Tools for PowerShellCmdlet-Referenz.

`New-R53QueryLoggingConfig`

Das folgende Codebeispiel zeigt die Verwendung. `New-R53QueryLoggingConfig`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine neue Route53-Konfiguration für die DNS-Abfrageprotokollierung für die angegebene Hosting-Zone erstellt. Amazon Route53 veröffentlicht DNS-Abfrageprotokolle in der angegebenen Cloudwatch-Protokollgruppe.

```
New-R53QueryLoggingConfig -HostedZoneId Z3MEQ8T7HAAAAF -CloudWatchLogsLogGroupArn
 arn:aws:logs:us-east-1:111111111111:log-group:/aws/route53/example.com:*
```

Ausgabe:

```
QueryLoggingConfig                      Location
------------------                      --------
Amazon.Route53.Model.QueryLoggingConfig https://route53.amazonaws.com/2013-04-01/
queryloggingconfig/ee5aaa95-4e03-4fdc-9d28-9e24ddaaaaa3
```

- Einzelheiten zur API finden Sie unter CreateQueryLoggingConfigCmdlet-Referenz.AWS Tools for PowerShell

`New-R53ReusableDelegationSet`

Das folgende Codebeispiel zeigt die Verwendung. `New-R53ReusableDelegationSet`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein wiederverwendbarer Delegierungssatz mit 4 Nameservern erstellt, die von mehreren gehosteten Zonen wiederverwendet werden können.

```
New-R53ReusableDelegationSet -CallerReference testcallerreference
```

Ausgabe:

```
DelegationSet                     Location
-------------                     --------
Amazon.Route53.Model.DelegationSet https://route53.amazonaws.com/2013-04-01/
delegationset/N23DS9XAAAAAXM
```

- Einzelheiten zur API finden Sie unter CreateReusableDelegationSet AWS Tools for PowerShellCmdlet-Referenz.

`Register-R53VPCWithHostedZone`

Das folgende Codebeispiel zeigt die Verwendung. `Register-R53VPCWithHostedZone`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene VPC der privaten Hosting-Zone zugeordnet.

```
Register-R53VPCWithHostedZone -HostedZoneId Z3MEQ8T7HAAAAF -VPC_VPCId vpc-f1b9aaaa -
VPC_VPCRegion us-east-1
```

Ausgabe:

```
Id                    Status  SubmittedAt         Comment
--                    ------  -----------         -------
/change/C3SCAAA633Z6DX PENDING 01/28/2020 19:32:02
```

- Einzelheiten zur API finden Sie unter AssociateVPC WithHostedZone in der Cmdlet-Referenz.AWS Tools for PowerShell

`Remove-R53HostedZone`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-R53HostedZone`

Tools für PowerShell

Beispiel 1: Löscht die gehostete Zone mit der angegebenen ID. Sie werden zur Bestätigung aufgefordert, bevor der Befehl ausgeführt wird, sofern Sie nicht den Switch-Parameter -Force hinzufügen.

```
Remove-R53HostedZone -Id Z1PA6795UKMFR9
```

- Einzelheiten zur API finden Sie unter DeleteHostedZone AWS Tools for PowerShellCmdlet-Referenz.

`Remove-R53QueryLoggingConfig`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-R53QueryLoggingConfig`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Konfiguration für die DNS-Abfrageprotokollierung entfernt.

```
Remove-R53QueryLoggingConfig -Id ee528e95-4e03-4fdc-9d28-9e24daaa20063
```

- Einzelheiten zur API finden Sie unter DeleteQueryLoggingConfig AWS Tools for PowerShellCmdlet-Referenz.

`Remove-R53ReusableDelegationSet`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-R53ReusableDelegationSet`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene wiederverwendbare Delegierungssatz gelöscht.

```
Remove-R53ReusableDelegationSet -Id N23DS9X4AYAAAM
```

- Einzelheiten zur API finden Sie unter DeleteReusableDelegationSet AWS Tools for PowerShellCmdlet-Referenz.

`Unregister-R53VPCFromHostedZone`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-R53VPCFromHostedZone`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene VPC von der privaten Hosting-Zone getrennt.

```
Unregister-R53VPCFromHostedZone -HostedZoneId Z3MEQ8T7HAAAAF -VPC_VPCId vpc-f1b9aaaa
 -VPC_VPCRegion us-east-1
```

Ausgabe:

```
Id                  Status  SubmittedAt         Comment
--                  ------  -----------         -------
```

```
/change/C2XFCAAAA9HKZG PENDING 01/28/2020 10:35:55
```

- Einzelheiten zur API finden Sie unter FromHostedZoneDisassociateVPC in der Cmdlet-Referenz.AWS Tools for PowerShell

`Update-R53HostedZoneComment`

Das folgende Codebeispiel zeigt die Verwendung. `Update-R53HostedZoneComment`

Tools für PowerShell

Beispiel 1: Dieser Befehl aktualisiert den Kommentar für die angegebene Hosting-Zone.

```
Update-R53HostedZoneComment -Id Z385PDS6AAAAAR -Comment "This is my first hosted
 zone"
```

Ausgabe:

```
Id                    : /hostedzone/Z385PDS6AAAAAR
Name                  : example.com.
CallerReference       : C5B55555-7147-EF04-8341-69131E805C89
Config                : Amazon.Route53.Model.HostedZoneConfig
ResourceRecordSetCount : 9
LinkedService         :
```

- Einzelheiten zur API finden Sie unter UpdateHostedZoneComment AWS Tools for PowerShellCmdlet-Referenz.

Amazon S3 S3-Beispiele mit Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon S3 Aktionen ausführen und allgemeine Szenarien implementieren. AWS Tools for PowerShell

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

Aktionen

`Copy-S3Object`

Das folgende Codebeispiel zeigt die Verwendung`Copy-S3Object`.

Tools für PowerShell

Beispiel 1: Dieser Befehl kopiert das Objekt "sample.txt" aus dem Bucket „test-files“ in denselben Bucket, jedoch mit dem neuen Schlüssel "sample-copy.txt“.

```
Copy-S3Object -BucketName test-files -Key sample.txt -DestinationKey sample-copy.txt
```

Beispiel 2: Dieser Befehl kopiert das Objekt "sample.txt" aus dem Bucket „test-files“ in den Bucket „backup-files“ mit dem Schlüssel "sample-copy.txt“.

```
Copy-S3Object -BucketName test-files -Key sample.txt -DestinationKey sample-copy.txt
 -DestinationBucket backup-files
```

Beispiel 3: Dieser Befehl lädt das Objekt "sample.txt" aus dem Bucket „test-files“ in eine lokale Datei mit dem Namen "local-sample.txt" herunter.

```
Copy-S3Object -BucketName test-files -Key sample.txt -LocalFile local-sample.txt
```

Beispiel 4: Lädt das einzelne Objekt in die angegebene Datei herunter. Die heruntergeladene Datei befindet sich unter c:\downloads\data\archive.zip

```
Copy-S3Object -BucketName test-files -Key data/archive.zip -LocalFolder c:\downloads
```

Beispiel 5: Lädt alle Objekte, die dem angegebenen key prefix entsprechen, in den lokalen Ordner herunter. Die relative Schlüsselhierarchie wird als Unterordner im gesamten Download-Speicherort beibehalten.

```
Copy-S3Object -BucketName test-files -KeyPrefix data -LocalFolder c:\downloads
```

- Einzelheiten zur API finden Sie unter CopyObject AWS Tools for PowerShellCmdlet-Referenz.

`Get-S3ACL`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3ACL`

Tools für PowerShell

Beispiel 1: Der Befehl ruft die Details des Objekteigentümers des S3-Objekts ab.

```
Get-S3ACL -BucketName 's3casetestbucket' -key 'initialize.ps1' -Select
 AccessControlList.Owner
```

Ausgabe:

```
DisplayName Id
----------- --
testusername      9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544
```

- Einzelheiten zur API finden Sie unter GetACL in der AWS Tools for PowerShell Cmdlet-Referenz.

`Get-S3Bucket`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3Bucket`

Tools für PowerShell

Beispiel 1: Dieser Befehl gibt alle S3-Buckets zurück.

```
Get-S3Bucket
```

Beispiel 2: Dieser Befehl gibt einen Bucket mit dem Namen „test-files“ zurück

```
Get-S3Bucket -BucketName test-files
```

- Einzelheiten zur API finden Sie unter ListBuckets AWS Tools for PowerShellCmdlet-Referenz.

`Get-S3BucketAccelerateConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketAccelerateConfiguration`

Tools für PowerShell

Beispiel 1: Dieser Befehl gibt den Wert Enabled zurück, wenn die Einstellungen für die Übertragungsbeschleunigung für den angegebenen Bucket aktiviert sind.

```
Get-S3BucketAccelerateConfiguration -BucketName 's3testbucket'
```

Ausgabe:

```
Value
-----
Enabled
```

- Einzelheiten zur API finden Sie unter GetBucketAccelerateConfiguration AWS Tools for PowerShellCmdlet-Referenz.

`Get-S3BucketAnalyticsConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketAnalyticsConfiguration`

Tools für PowerShell

Beispiel 1: Dieser Befehl gibt die Details des Analysefilters mit dem Namen 'testfilter' im angegebenen S3-Bucket zurück.

```
Get-S3BucketAnalyticsConfiguration -BucketName 's3testbucket' -AnalyticsId
 'testfilter'
```

- Einzelheiten zur API finden Sie unter GetBucketAnalyticsConfiguration AWS Tools for PowerShellCmdlet-Referenz.

`Get-S3BucketAnalyticsConfigurationList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketAnalyticsConfigurationList`

Tools für PowerShell

Beispiel 1: Dieser Befehl gibt die ersten 100 Analytics-Konfigurationen des angegebenen S3-Buckets zurück.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 's3casetestbucket'
```

- Einzelheiten zur API finden Sie unter ListBucketAnalyticsConfigurations AWS Tools for PowerShellCmdlet-Referenz.

`Get-S3BucketEncryption`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketEncryption`

Tools für PowerShell

Beispiel 1: Dieser Befehl gibt alle serverseitigen Verschlüsselungsregeln zurück, die dem angegebenen Bucket zugeordnet sind.

```
Get-S3BucketEncryption -BucketName 's3casetestbucket'
```

- Einzelheiten zur API finden Sie unter GetBucketEncryption AWS Tools for PowerShellCmdlet-Referenz.

`Get-S3BucketInventoryConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketInventoryConfiguration`

Tools für PowerShell

Beispiel 1: Dieser Befehl gibt die Details des Inventars mit dem Namen 'testinventory' für den angegebenen S3-Bucket zurück.

```
Get-S3BucketInventoryConfiguration -BucketName 's3testbucket' -InventoryId
 'testinventory'
```

- Einzelheiten zur API finden Sie unter GetBucketInventoryConfiguration AWS Tools for PowerShellCmdlet-Referenz.

`Get-S3BucketInventoryConfigurationList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketInventoryConfigurationList`

Tools für PowerShell

Beispiel 1: Dieser Befehl gibt die ersten 100 Inventarkonfigurationen des angegebenen S3-Buckets zurück.

```
Get-S3BucketInventoryConfigurationList -BucketName 's3testbucket'
```

- Einzelheiten zur API finden Sie unter ListBucketInventoryConfigurations AWS Tools for PowerShellCmdlet-Referenz.

`Get-S3BucketLocation`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketLocation`

Tools für PowerShell

Beispiel 1: Dieser Befehl gibt die Standortbeschränkung für den Bucket 's3testbucket' zurück, falls eine Einschränkung existiert.

```
Get-S3BucketLocation -BucketName 's3testbucket'
```

Ausgabe:

```
Value
-----
ap-south-1
```

- Einzelheiten zur API finden Sie unter GetBucketLocationCmdlet-Referenz.AWS Tools for PowerShell

`Get-S3BucketLogging`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketLogging`

Tools für PowerShell

Beispiel 1: Dieser Befehl gibt den Logging-Status für den angegebenen Bucket zurück.

```
Get-S3BucketLogging -BucketName 's3testbucket'
```

Ausgabe:

```
TargetBucketName   Grants TargetPrefix
----------------   ------ ------------
testbucket1        {}     testprefix
```

- Einzelheiten zur API finden Sie unter GetBucketLogging AWS Tools for PowerShellCmdlet-Referenz.

`Get-S3BucketMetricsConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketMetricsConfiguration`

Tools für PowerShell

Beispiel 1: Dieser Befehl gibt die Details zum Metrikfilter mit dem Namen 'testfilter' für den angegebenen S3-Bucket zurück.

```
Get-S3BucketMetricsConfiguration -BucketName 's3testbucket' -MetricsId 'testfilter'
```

- Einzelheiten zur API finden Sie unter GetBucketMetricsConfiguration AWS Tools for PowerShellCmdlet-Referenz.

`Get-S3BucketNotification`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketNotification`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Benachrichtigungskonfiguration des angegebenen Buckets abgerufen

```
Get-S3BucketNotification -BucketName kt-tools | select -ExpandProperty
 TopicConfigurations
```

Ausgabe:

```
Id   Topic
--   -----
mimo arn:aws:sns:eu-west-1:123456789012:topic-1
```

- Einzelheiten zur API finden Sie unter GetBucketNotification AWS Tools for PowerShellCmdlet-Referenz.

`Get-S3BucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketPolicy`

Tools für PowerShell

Beispiel 1: Dieser Befehl gibt die Bucket-Richtlinie aus, die dem angegebenen S3-Bucket zugeordnet ist.

```
Get-S3BucketPolicy -BucketName 's3testbucket'
```

- Einzelheiten zur API finden Sie unter GetBucketPolicy AWS Tools for PowerShellCmdlet-Referenz.

`Get-S3BucketPolicyStatus`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketPolicyStatus`

Tools für PowerShell

Beispiel 1: Dieser Befehl gibt den Richtlinienstatus für den angegebenen S3-Bucket zurück und gibt an, ob der Bucket öffentlich ist.

```
Get-S3BucketPolicyStatus -BucketName 's3casetestbucket'
```

- Einzelheiten zur API finden Sie unter GetBucketPolicyStatus AWS Tools for PowerShellCmdlet-Referenz.

`Get-S3BucketReplication`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketReplication`

Tools für PowerShell

Beispiel 1: Gibt die Informationen zur Replikationskonfiguration zurück, die für den Bucket mit dem Namen „mybucket“ festgelegt wurden.

```
Get-S3BucketReplication -BucketName mybucket
```

- Einzelheiten zur API finden Sie unter GetBucketReplicationCmdlet-Referenz.AWS Tools for PowerShell

`Get-S3BucketRequestPayment`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketRequestPayment`

Tools für PowerShell

Beispiel 1: Gibt die Konfiguration der Anforderungszahlung für den Bucket mit dem Namen „mybucket“ zurück. Standardmäßig zahlt der Bucket-Besitzer für Downloads aus dem Bucket.

```
Get-S3BucketRequestPayment -BucketName mybucket
```

- Einzelheiten zur API finden Sie unter GetBucketRequestPayment AWS Tools for PowerShellCmdlet-Referenz.

`Get-S3BucketTagging`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketTagging`

Tools für PowerShell

Beispiel 1: Dieser Befehl gibt alle Tags zurück, die dem angegebenen Bucket zugeordnet sind.

```
Get-S3BucketTagging -BucketName 's3casetestbucket'
```

- Einzelheiten zur API finden Sie unter GetBucketTagging AWS Tools for PowerShellCmdlet-Referenz.

`Get-S3BucketVersioning`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketVersioning`

Tools für PowerShell

Beispiel 1: Dieser Befehl gibt den Status der Versionierung in Bezug auf den angegebenen Bucket zurück.

```
Get-S3BucketVersioning -BucketName 's3testbucket'
```

- Einzelheiten zur API finden Sie unter GetBucketVersioning AWS Tools for PowerShellCmdlet-Referenz.

`Get-S3BucketWebsite`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketWebsite`

Tools für PowerShell

Beispiel 1: Dieser Befehl gibt die Details der statischen Website-Konfigurationen des angegebenen S3-Buckets zurück.

```
Get-S3BucketWebsite -BucketName 's3testbucket'
```

- Einzelheiten zur API finden Sie unter GetBucketWebsite AWS Tools for PowerShellCmdlet-Referenz.

`Get-S3CORSConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3CORSConfiguration`

Tools für PowerShell

Beispiel 1: Dieser Befehl gibt ein Objekt zurück, das alle CORS-Konfigurationsregeln enthält, die dem angegebenen S3-Bucket entsprechen.

```
Get-S3CORSConfiguration -BucketName 's3testbucket' -Select Configuration.Rules
```

Ausgabe:

```
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 : {}
```

- API-Details finden Sie unter GetCorsConfiguration in der Cmdlet-Referenz.AWS Tools for PowerShell

`Get-S3LifecycleConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3LifecycleConfiguration`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Lebenszykluskonfiguration für den Bucket abgerufen.

```
Get-S3LifecycleConfiguration -BucketName test-bla
```

Ausgabe:

```
Rules
-----
{Remove-in-150-days, Archive-to-Glacier-in-30-days}
```

- Einzelheiten zur API finden Sie unter GetLifecycleConfiguration AWS Tools for PowerShellCmdlet-Referenz.

`Get-S3Object`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3Object`

Tools für PowerShell

Beispiel 1: Dieser Befehl ruft die Informationen über alle Elemente im Bucket „test-files“ ab.

```
Get-S3Object -BucketName test-files
```

Beispiel 2: Dieser Befehl ruft die Informationen über das Objekt "sample.txt" aus dem Bucket „test-files“ ab.

```
Get-S3Object -BucketName test-files -Key sample.txt
```

Beispiel 3: Dieser Befehl ruft die Informationen über alle Elemente mit dem Präfix „sample“ aus dem Bucket „test-files“ ab.

```
Get-S3Object -BucketName test-files -KeyPrefix sample
```

- Einzelheiten zur API finden Sie unter ListObjectsCmdlet-Referenz.AWS Tools for PowerShell

`Get-S3ObjectLockConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3ObjectLockConfiguration`

Tools für PowerShell

Beispiel 1: Dieser Befehl gibt den Wert 'Enabled' zurück, wenn die Objektsperrkonfiguration für den angegebenen S3-Bucket aktiviert ist.

```
Get-S3ObjectLockConfiguration -BucketName 's3buckettesting' -Select
 ObjectLockConfiguration.ObjectLockEnabled
```

Ausgabe:

```
Value
-----
```

```
Enabled
```

- Einzelheiten zur API finden Sie unter GetObjectLockConfiguration AWS Tools for PowerShellCmdlet-Referenz.

`Get-S3ObjectMetadata`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3ObjectMetadata`

Tools für PowerShell

Beispiel 1: Dieser Befehl gibt die Metadaten des Objekts mit dem Schlüssel 'ListTrusts.txt' im angegebenen S3-Bucket zurück.

```
Get-S3ObjectMetadata -BucketName 's3testbucket' -Key 'ListTrusts.txt'
```

Ausgabe:

```
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                               :
```

- Einzelheiten zur API finden Sie unter GetObjectMetadata AWS Tools for PowerShellCmdlet-Referenz.

`Get-S3ObjectRetention`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3ObjectRetention`

Tools für PowerShell

Beispiel 1: Der Befehl gibt den Modus und das Datum zurück, bis das Objekt beibehalten werden würde.

```
Get-S3ObjectRetention -BucketName 's3buckettesting' -Key 'testfile.txt'
```

- Einzelheiten zur API finden Sie unter GetObjectRetention AWS Tools for PowerShellCmdlet-Referenz.

`Get-S3ObjectTagSet`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3ObjectTagSet`

Tools für PowerShell

Beispiel 1: Das Beispiel gibt die Tags zurück, die dem Objekt zugeordnet sind, das im angegebenen S3-Bucket vorhanden ist.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'testbucket123'
```

Ausgabe:

```
Key  Value
---  -----
test value
```

- Einzelheiten zur API finden Sie unter GetObjectTagging AWS Tools for PowerShellCmdlet-Referenz.

`Get-S3PreSignedURL`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3PreSignedURL`

Tools für PowerShell

Beispiel 1: Der Befehl gibt eine vorsignierte URL für einen angegebenen Schlüssel und ein Ablaufdatum zurück.

```
Get-S3PreSignedURL -BucketName 's3testbucket' -Key 'testkey' -Expires '2023-11-16'
```

Beispiel 2: Der Befehl gibt eine vorsignierte URL für einen Directory-Bucket mit dem angegebenen Schlüssel und einem Ablaufdatum zurück.

```
[Amazon.AWSConfigsS3]::UseSignatureVersion4 = $true
      Get-S3PreSignedURL -BucketName sampledirectorybucket--use1-az5--x-s3 -Key
 'testkey' -Expire '2023-11-17'
```

- Einzelheiten zur API finden Sie unter GetPreSignedURL in der AWS Tools for PowerShell Cmdlet-Referenz.

`Get-S3PublicAccessBlock`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3PublicAccessBlock`

Tools für PowerShell

Beispiel 1: Der Befehl gibt die Blockkonfiguration für den öffentlichen Zugriff des angegebenen S3-Buckets zurück.

```
Get-S3PublicAccessBlock -BucketName 's3testbucket'
```

- Einzelheiten zur API finden Sie unter GetPublicAccessBlock AWS Tools for PowerShellCmdlet-Referenz.

`Get-S3Version`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3Version`

Tools für PowerShell

Beispiel 1: Dieser Befehl gibt die Metadaten zu allen Versionen von Objekten im angegebenen S3-Bucket zurück.

```
Get-S3Version -BucketName 's3testbucket'
```

Ausgabe:

```
IsTruncated         : False
KeyMarker           :
VersionIdMarker     :
NextKeyMarker       :
NextVersionIdMarker :
Versions            : {EC2.txt, EC2MicrosoftWindowsGuide.txt, ListDirectories.json,
 ListTrusts.json}
Name                : s3testbucket
Prefix              :
MaxKeys             : 1000
CommonPrefixes      : {}
Delimiter           :
```

- Einzelheiten zur API finden Sie unter ListVersions AWS Tools for PowerShellCmdlet-Referenz.

`New-S3Bucket`

Das folgende Codebeispiel zeigt die Verwendung. `New-S3Bucket`

Tools für PowerShell

Beispiel 1: Dieser Befehl erstellt einen neuen privaten Bucket mit dem Namen „sample-bucket“.

```
New-S3Bucket -BucketName sample-bucket
```

Beispiel 2: Dieser Befehl erstellt einen neuen Bucket mit dem Namen „sample-bucket“ mit Lese- und Schreibberechtigungen.

```
New-S3Bucket -BucketName sample-bucket -PublicReadWrite
```

Beispiel 3: Dieser Befehl erstellt einen neuen Bucket mit dem Namen „sample-bucket“ mit schreibgeschützten Rechten.

```
New-S3Bucket -BucketName sample-bucket -PublicReadOnly
```

Beispiel 4: Dieser Befehl erstellt einen neuen Verzeichnis-Bucket mit dem Namen „samplebucket--use1-az5--x-s3" mit. PutBucketConfiguration

```
$bucketConfiguration = @{
      BucketInfo = @{
          DataRedundancy = 'SingleAvailabilityZone'
          Type = 'Directory'
      }
      Location = @{
          Name = 'use1-az5'
          Type = 'AvailabilityZone'
      }
    }
New-S3Bucket -BucketName samplebucket--use1-az5--x-s3 -BucketConfiguration
 $bucketConfiguration -Region us-east-1
```

- Einzelheiten zur PutBucketAPI AWS Tools for PowerShell finden Sie unter Cmdlet-Referenz.

`Read-S3Object`

Das folgende Codebeispiel zeigt die Verwendung. `Read-S3Object`

Tools für PowerShell

Beispiel 1: Dieser Befehl ruft das Element "sample.txt" aus dem Bucket „test-files“ ab und speichert es in einer Datei mit dem Namen "local-sample.txt" am aktuellen Speicherort. Die Datei "local-sample.txt" muss nicht existieren, bevor dieser Befehl aufgerufen wird.

```
Read-S3Object -BucketName test-files -Key sample.txt -File local-sample.txt
```

Beispiel 2: Dieser Befehl ruft das virtuelle Verzeichnis „DIR“ aus dem Bucket „test-files“ ab und speichert es in einem Ordner mit dem Namen „Local-dir“ am aktuellen Speicherort. Der Ordner „Local-dir“ muss nicht existieren, bevor dieser Befehl aufgerufen wird.

```
Read-S3Object -BucketName test-files -KeyPrefix DIR -Folder Local-DIR
```

Beispiel 3: Lädt alle Objekte mit Schlüsseln, die auf '.json' enden, aus Buckets mit 'config' im Bucket-Namen in Dateien im angegebenen Ordner herunter. Die Objektschlüssel werden verwendet, um die Dateinamen festzulegen.

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -
like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

- Einzelheiten zur API finden Sie unter GetObject AWS Tools for PowerShellCmdlet-Referenz.

`Remove-S3Bucket`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3Bucket`

Tools für PowerShell

Beispiel 1: Dieser Befehl entfernt alle Objekte und Objektversionen aus dem Bucket 'test-files' und löscht dann den Bucket. Der Befehl fordert Sie zur Bestätigung auf, bevor Sie fortfahren. Fügen Sie den Schalter -Force hinzu, um die Bestätigung zu unterdrücken. Beachten Sie, dass Buckets, die nicht leer sind, nicht gelöscht werden können.

```
Remove-S3Bucket -BucketName test-files -DeleteBucketContent
```

- Einzelheiten zur API finden Sie unter DeleteBucket AWS Tools for PowerShellCmdlet-Referenz.

`Remove-S3BucketAnalyticsConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3BucketAnalyticsConfiguration`

Tools für PowerShell

Beispiel 1: Der Befehl entfernt den Analysefilter mit dem Namen 'testfilter' aus dem angegebenen S3-Bucket.

```
Remove-S3BucketAnalyticsConfiguration -BucketName 's3testbucket' -AnalyticsId
 'testfilter'
```

- Einzelheiten zur API finden Sie unter DeleteBucketAnalyticsConfiguration AWS Tools for PowerShellCmdlet-Referenz.

`Remove-S3BucketEncryption`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3BucketEncryption`

Tools für PowerShell

Beispiel 1: Dadurch wird die für den bereitgestellten S3-Bucket aktivierte Verschlüsselung deaktiviert.

```
Remove-S3BucketEncryption -BucketName 's3casetestbucket'
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DeleteBucketEncryption AWS Tools for PowerShellCmdlet-Referenz.

`Remove-S3BucketInventoryConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3BucketInventoryConfiguration`

Tools für PowerShell

Beispiel 1: Dieser Befehl entfernt das Inventar mit dem Namen 'testInventoryName', das dem angegebenen S3-Bucket entspricht.

```
Remove-S3BucketInventoryConfiguration -BucketName 's3testbucket' -InventoryId
 'testInventoryName'
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DeleteBucketInventoryConfiguration AWS Tools for PowerShellCmdlet-Referenz.

`Remove-S3BucketMetricsConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3BucketMetricsConfiguration`

Tools für PowerShell

Beispiel 1: Der Befehl entfernt den Metrikfilter mit dem Namen 'testmetrics' im angegebenen S3-Bucket.

```
Remove-S3BucketMetricsConfiguration -BucketName 's3testbucket' -MetricsId
 'testmetrics'
```

- Einzelheiten zur API finden Sie unter DeleteBucketMetricsConfiguration AWS Tools for PowerShellCmdlet-Referenz.

`Remove-S3BucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3BucketPolicy`

Tools für PowerShell

Beispiel 1: Der Befehl entfernt die Bucket-Richtlinie, die dem angegebenen S3-Bucket zugeordnet ist.

```
Remove-S3BucketPolicy -BucketName 's3testbucket'
```

- Einzelheiten zur API finden Sie unter DeleteBucketPolicy AWS Tools for PowerShellCmdlet-Referenz.

`Remove-S3BucketReplication`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3BucketReplication`

Tools für PowerShell

Beispiel 1: Löscht die Replikationskonfiguration, die dem Bucket mit dem Namen „mybucket“ zugeordnet ist. Beachten Sie, dass für diesen Vorgang eine Genehmigung für die Aktion s3:

erforderlich ist. DeleteReplicationConfiguration Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird. Um die Bestätigung zu unterdrücken, verwenden Sie den Schalter -Force.

```
Remove-S3BucketReplication -BucketName mybucket
```

- Einzelheiten zur API finden Sie unter DeleteBucketReplication AWS Tools for PowerShellCmdlet-Referenz.

`Remove-S3BucketTagging`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3BucketTagging`

Tools für PowerShell

Beispiel 1: Dieser Befehl entfernt alle Tags, die dem angegebenen S3-Bucket zugeordnet sind.

```
Remove-S3BucketTagging -BucketName 's3testbucket'
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DeleteBucketTagging AWS Tools for PowerShellCmdlet-Referenz.

`Remove-S3BucketWebsite`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3BucketWebsite`

Tools für PowerShell

Beispiel 1: Dieser Befehl deaktiviert die statische Website-Hosting-Eigenschaft des angegebenen S3-Buckets.

```
Remove-S3BucketWebsite -BucketName 's3testbucket'
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DeleteBucketWebsite AWS Tools for PowerShellCmdlet-Referenz.

`Remove-S3CORSConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3CORSConfiguration`

Tools für PowerShell

Beispiel 1: Dieser Befehl entfernt die CORS-Konfiguration für den angegebenen S3-Bucket.

```
Remove-S3CORSConfiguration -BucketName 's3testbucket'
```

Ausgabe:

```
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
```

- API-Details finden Sie unter DeleteCorsConfiguration in der Cmdlet-Referenz.AWS Tools for PowerShell

`Remove-S3LifecycleConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3LifecycleConfiguration`

Tools für PowerShell

Beispiel 1: Der Befehl entfernt alle Lebenszyklusregeln für den angegebenen S3-Bucket.

```
Remove-S3LifecycleConfiguration -BucketName 's3testbucket'
```

- Einzelheiten zur API finden Sie unter DeleteLifecycleConfiguration AWS Tools for PowerShellCmdlet-Referenz.

`Remove-S3MultipartUpload`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3MultipartUpload`

Tools für PowerShell

Beispiel 1: Mit diesem Befehl werden mehrteilige Uploads abgebrochen, die vor mehr als 5 Tagen erstellt wurden.

```
Remove-S3MultipartUpload -BucketName test-files -DaysBefore 5
```

Beispiel 2: Mit diesem Befehl werden mehrteilige Uploads abgebrochen, die vor dem 2. Januar 2014 erstellt wurden.

```
Remove-S3MultipartUpload -BucketName test-files -InitiatedDate "Thursday, January
 02, 2014"
```

Beispiel 3: Mit diesem Befehl werden mehrteilige Uploads abgebrochen, die vor dem 2. Januar 2014, 10:45:37 erstellt wurden.

```
Remove-S3MultipartUpload -BucketName test-files -InitiatedDate "2014/01/02 10:45:37"
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz. AbortMultipartUploadAWS Tools for PowerShell

`Remove-S3Object`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3Object`

Beispiel 1: Dieser Befehl entfernt das Objekt "sample.txt" aus dem Bucket „test-files“. Sie werden zur Bestätigung aufgefordert, bevor der Befehl ausgeführt wird. Um die Aufforderung zu unterdrücken, verwenden Sie den Schalter -Force.

```
Remove-S3Object -BucketName test-files -Key sample.txt
```

Beispiel 2: Dieser Befehl entfernt die angegebene Version des Objekts "sample.txt" aus dem Bucket „test-files“, vorausgesetzt, der Bucket wurde so konfiguriert, dass Objektversionen aktiviert werden.

```
Remove-S3Object -BucketName test-files -Key sample.txt -VersionId
 HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

Beispiel 3: Dieser Befehl entfernt die Objekte "sample1.txt „," sample2.txt "und" sample3.txt "aus dem Bucket „test-files“ als einzelne Batch-Operation. In der Serviceantwort werden alle verarbeiteten Schlüssel aufgeführt, unabhängig vom Erfolgs- oder Fehlerstatus des Löschvorgangs. Um nur Fehler für Schlüssel zu erhalten, die vom Dienst nicht verarbeitet werden konnten, fügen Sie den ReportErrorsOnly Parameter - hinzu (dieser Parameter kann auch mit dem Alias -Quiet angegeben werden).

```
Remove-S3Object -BucketName test-files -KeyCollection @( "sample1.txt",
 "sample2.txt", "sample3.txt" )
```

Beispiel 4: In diesem Beispiel wird ein Inline-Ausdruck mit dem KeyCollection Parameter - verwendet, um die Schlüssel der zu löschenden Objekte abzurufen. Get-S3Objectgibt eine Sammlung von Amazon.S3.Model.S3Object-Instanzen zurück, von denen jede ein Key-Element vom Typ Zeichenfolge hat, das das Objekt identifiziert.

```
Remove-S3Object -bucketname "test-files" -KeyCollection (Get-S3Object "test-files" -
KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

Beispiel 5: In diesem Beispiel werden alle Objekte abgerufen, die ein key prefix „Präfix/Unterpräfix“ im Bucket haben, und sie werden gelöscht. Beachten Sie, dass die eingehenden Objekte nacheinander verarbeitet werden. Bei großen Sammlungen sollten Sie erwägen, die Sammlung an den Parameter - InputObject (alias -S3ObjectCollection) des Cmdlets zu übergeben, damit das Löschen als Batch mit einem einzigen Aufruf des Dienstes erfolgen kann.

```
Get-S3Object -BucketName "test-files" -KeyPrefix "prefix/subprefix" | Remove-
S3Object -Force
```

Beispiel 6: In diesem Beispiel wird eine Sammlung von ObjectVersion Amazon.S3.Model.S3-Instances, die Löschmarkierungen darstellen, zur Löschung an das Cmdlet übergeben. Beachten Sie, dass die eingehenden Objekte nacheinander verarbeitet werden. Bei großen Sammlungen sollten Sie erwägen, die Sammlung an den Parameter - InputObject (alias -S3ObjectCollection) des Cmdlets zu übergeben, damit das Löschen als Batch mit einem einzigen Aufruf des Dienstes erfolgen kann.

```
(Get-S3Version -BucketName "test-files").Versions | Where {$_.IsDeleteMarker -eq
 "True"} | Remove-S3Object -Force
```

Beispiel 7: Dieses Skript zeigt, wie eine Gruppe von Objekten (in diesem Fall Löschmarken) im Batch-Modus gelöscht werden kann, indem ein Array von Objekten erstellt wird, die mit dem Parameter - verwendet werden sollen. KeyAndVersionCollection

```
$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
```

- Einzelheiten zur API finden Sie unter DeleteObjects AWS Tools for PowerShellCmdlet-Referenz.

`Remove-S3ObjectTagSet`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3ObjectTagSet`

Tools für PowerShell

Beispiel 1: Dieser Befehl entfernt alle Tags, die dem Objekt mit dem Schlüssel 'testfile.txt' im angegebenen S3-Bucket zugeordnet sind.

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 's3testbucket' -Select '^Key'
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DeleteObjectTagging AWS Tools for PowerShellCmdlet-Referenz.

`Remove-S3PublicAccessBlock`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3PublicAccessBlock`

Tools für PowerShell

Beispiel 1: Dieser Befehl deaktiviert die Einstellung „Öffentlichen Zugriff blockieren“ für den angegebenen Bucket.

```
Remove-S3PublicAccessBlock -BucketName 's3testbucket' -Force -Select '^BucketName'
```

Ausgabe:

```
s3testbucket
```

- Einzelheiten zur API finden Sie unter DeletePublicAccessBlock AWS Tools for PowerShellCmdlet-Referenz.

`Set-S3BucketEncryption`

Das folgende Codebeispiel zeigt die Verwendung. `Set-S3BucketEncryption`

Tools für PowerShell

Beispiel 1: Dieser Befehl aktiviert die serverseitige AES256-Standardverschlüsselung mit Amazon S3 Managed Keys (SSE-S3) für den angegebenen Bucket.

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =
 @{ServerSideEncryptionAlgorithm = "AES256"}}
```

```
Set-S3BucketEncryption -BucketName 's3testbucket' -
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- Einzelheiten zur API finden Sie unter PutBucketEncryptionCmdlet-Referenz.AWS Tools for PowerShell

`Test-S3Bucket`

Das folgende Codebeispiel zeigt die Verwendung. `Test-S3Bucket`

Tools für PowerShell

Beispiel 1: Dieser Befehl gibt True zurück, wenn der Bucket existiert, andernfalls False. Der Befehl gibt True zurück, auch wenn der Bucket nicht dem Benutzer gehört.

```
Test-S3Bucket -BucketName test-files
```

- Einzelheiten zur API finden Sie unter Test-S3Bucket AWS Tools for PowerShellCmdlet-Referenz.

`Write-S3BucketAccelerateConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Write-S3BucketAccelerateConfiguration`

Tools für PowerShell

Beispiel 1: Dieser Befehl aktiviert die Übertragungsbeschleunigung für den angegebenen S3-Bucket.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')
Write-S3BucketAccelerateConfiguration -BucketName 's3testbucket' -
AccelerateConfiguration_Status $statusVal
```

- Einzelheiten zur API finden Sie unter PutBucketAccelerateConfiguration AWS Tools for PowerShellCmdlet-Referenz.

`Write-S3BucketNotification`

Das folgende Codebeispiel zeigt die Verwendung. `Write-S3BucketNotification`

Beispiel 1: In diesem Beispiel wird die SNS-Themenkonfiguration für das S3-Ereignis konfiguriert ObjectRemovedDelete und die Benachrichtigung für den angegebenen S3-Bucket aktiviert

```
$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 kt-tools -TopicConfiguration $topic
```

Beispiel 2: Dieses Beispiel aktiviert Benachrichtigungen ObjectCreatedAll für den angegebenen Bucket und sendet ihn an die Lambda-Funktion.

```
$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 ssm-editor -LambdaFunctionConfiguration
 $lambdaConfig
```

Beispiel 3: In diesem Beispiel werden zwei verschiedene Lambda-Konfigurationen auf der Grundlage unterschiedlicher Schlüsselsuffixe erstellt und beide in einem einzigen Befehl konfiguriert.

```
#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 ssm-editor -LambdaFunctionConfiguration
 $firstLambdaConfig,$secondlambdaConfig
```

- Einzelheiten zur API finden Sie unter PutBucketNotificationCmdlet-Referenz.AWS Tools for PowerShell

`Write-S3BucketReplication`

Das folgende Codebeispiel zeigt die Verwendung. `Write-S3BucketReplication`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Replikationskonfiguration mit einer einzigen Regel eingerichtet, die die Replikation aller neuen Objekte, die mit dem Schlüsselnamenpräfix

"TaxDocs" im Bucket 'examplebucket' erstellt wurden, in den Bucket 'exampletargetbucket' ermöglicht.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::exampletargetbucket" }

$params = @{
    BucketName = "examplebucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

Beispiel 2: In diesem Beispiel wird eine Replikationskonfiguration mit mehreren Regeln festgelegt, die die Replikation aller neuen Objekte, die entweder mit dem Schlüsselnamenpräfix "" oder "" erstellt wurden, in den Bucket 'exampletargetbucket' ermöglichen. TaxDocs OtherDocs Die Schlüsselpräfixe dürfen sich nicht überschneiden.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::exampletargetbucket" }

$rule2 = New-Object Amazon.S3.Model.ReplicationRule
$rule2.ID = "Rule-2"
$rule2.Status = "Enabled"
$rule2.Prefix = "OtherDocs"
$rule2.Destination = @{ BucketArn = "arn:aws:s3:::exampletargetbucket" }

$params = @{
    BucketName = "examplebucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1,$rule2
}
```

```
Write-S3BucketReplication @params
```

Beispiel 3: In diesem Beispiel wird die Replikationskonfiguration für den angegebenen Bucket aktualisiert, um die Regel zu deaktivieren, die die Replikation von Objekten mit dem Schlüsselnamenpräfix "TaxDocs" in den Bucket 'exampletargetbucket' steuert.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Disabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::exampletargetbucket" }

$params = @{
    BucketName = "examplebucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz. PutBucketReplicationAWS Tools for PowerShell

`Write-S3BucketRequestPayment`

Das folgende Codebeispiel zeigt die Verwendung. `Write-S3BucketRequestPayment`

Tools für PowerShell

Beispiel 1: Aktualisiert die Konfiguration der Zahlungsanfrage für den Bucket mit dem Namen „mybucket“, sodass der Person, die Downloads aus dem Bucket anfordert, der Download in Rechnung gestellt wird. Standardmäßig zahlt der Bucket-Besitzer für Downloads. Verwenden Sie 'BucketOwner' für den Parameter RequestPaymentConfiguration _Payer, um die Zahlung für die Anfrage wieder auf die Standardwerte zurückzusetzen.

```
Write-S3BucketRequestPayment -BucketName mybucket -RequestPaymentConfiguration_Payer
 Requester
```

- Einzelheiten zur API finden Sie unter PutBucketRequestPayment AWS Tools for PowerShellCmdlet-Referenz.

`Write-S3BucketTagging`

Das folgende Codebeispiel zeigt die Verwendung. `Write-S3BucketTagging`

Tools für PowerShell

Beispiel 1: Dieser Befehl wendet zwei Tags auf einen Bucket mit dem Namen an**`cloudtrail-test-2018`**: ein Tag mit dem Schlüssel Stage und dem Wert Test und ein Tag mit dem Schlüssel Environment und dem Wert Alpha. Führen Sie den Befehl aus, um zu überprüfen, ob die Tags dem Bucket hinzugefügt wurden**`Get-S3BucketTagging -BucketName bucket_name`**. Die Ergebnisse sollten die Tags enthalten, die Sie im ersten Befehl auf den Bucket angewendet haben. Beachten Sie, dass **`Write-S3BucketTagging`** dadurch der gesamte vorhandene Tagsatz in einem Bucket überschrieben wird. Um einzelne Tags hinzuzufügen oder zu löschen, führen Sie die API-Cmdlets Resource Groups und Tagging aus. **`Add-RGTResourceTag`** **`Remove-RGTResourceTag`** Verwenden Sie alternativ den Tag-Editor in der AWS Management Console, um S3-Bucket-Tags zu verwalten.

```
Write-S3BucketTagging -BucketName cloudtrail-test-2018 -TagSet @( @{ Key="Stage";
 Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

Beispiel 2: Mit diesem Befehl wird ein Bucket mit dem Namen an **`cloudtrail-test-2018`** das **`Write-S3BucketTagging`** Cmdlet übergeben. Er wendet die Tags Stage:Production und Department:Finance auf den Bucket an. Beachten Sie, dass dadurch der gesamte vorhandene Tagsatz in einem **`Write-S3BucketTagging`** Bucket überschrieben wird.

```
Get-S3Bucket -BucketName cloudtrail-test-2018 | Write-S3BucketTagging -TagSet
 @( @{ Key="Stage"; Value="Production" }, @{ Key="Department"; Value="Finance" } )
```

- Einzelheiten zur API finden Sie unter PutBucketTagging AWS Tools for PowerShellCmdlet-Referenz.

`Write-S3BucketVersioning`

Das folgende Codebeispiel zeigt die Verwendung. `Write-S3BucketVersioning`

Tools für PowerShell

Beispiel 1: Der Befehl aktiviert die Versionierung für den angegebenen S3-Bucket.

```
Write-S3BucketVersioning -BucketName 's3testbucket' -VersioningConfig_Status Enabled
```

- Einzelheiten zur API finden Sie unter PutBucketVersioning AWS Tools for PowerShellCmdlet-Referenz.

`Write-S3BucketWebsite`

Das folgende Codebeispiel zeigt die Verwendung. `Write-S3BucketWebsite`

Tools für PowerShell

Beispiel 1: Der Befehl aktiviert das Website-Hosting für den angegebenen Bucket mit dem Indexdokument als 'index.html' und dem Fehlerdokument als 'error.html'.

```
Write-S3BucketWebsite -BucketName 's3testbucket' -
WebsiteConfiguration_IndexDocumentSuffix 'index.html' -
WebsiteConfiguration_ErrorDocument 'error.html'
```

- Einzelheiten zur API finden Sie unter PutBucketWebsiteCmdlet-Referenz.AWS Tools for PowerShell

`Write-S3LifecycleConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Write-S3LifecycleConfiguration`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die im $ angegebene Konfiguration geschrieben/ersetztNewRule. Diese Konfiguration stellt sicher, dass die Bereichsobjekte mit bestimmten Präfix- und Tag-Werten begrenzt werden.

```
$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 my-review-scrap -Configuration_Rule
$NewRule
```

Beispiel 2: In diesem Beispiel werden mehrere Regeln mit Filterung festgelegt. $ ArchiveRule legt fest, dass die Objekte in 30 Tagen auf Glacier und in 120 Tagen archiviert DeepArchive werden sollen. $ ExpireRule läuft sowohl in der aktuellen als auch in früheren Versionen in 150 Tagen für Objekte ab, bei denen das Präfix 'py' und der tag:key 'archieved' auf 'yes' gesetzt sind.

```
$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 my-review-scrap -Configuration_Rule
$ExpireRule,$ArchiveRule
```

- Einzelheiten zur API finden Sie in der Cmdlet-Referenz. PutLifecycleConfigurationAWS Tools for PowerShell

`Write-S3Object`

Das folgende Codebeispiel zeigt die Verwendung. `Write-S3Object`

Tools für PowerShell

Beispiel 1: Dieser Befehl lädt die einzelne Datei "local-sample.txt" auf Amazon S3 hoch und erstellt ein Objekt mit dem Schlüssel "sample.txt" im Bucket „test-files“.

```
Write-S3Object -BucketName test-files -Key "sample.txt" -File .\local-sample.txt
```

Beispiel 2: Dieser Befehl lädt die einzelne Datei "sample.txt" auf Amazon S3 hoch und erstellt ein Objekt mit dem Schlüssel "sample.txt" im Bucket „test-files“. Wenn der -Key-Parameter nicht angegeben wird, wird der Dateiname als S3-Objektschlüssel verwendet.

```
Write-S3Object -BucketName test-files -File .\sample.txt
```

Beispiel 3: Dieser Befehl lädt die einzelne Datei "local-sample.txt" auf Amazon S3 hoch und erstellt ein Objekt mit dem Schlüssel "prefix/to/sample.txt" im Bucket „test-files“.

```
Write-S3Object -BucketName test-files -Key "prefix/to/sample.txt" -File .\local-
sample.txt
```

Beispiel 4: Dieser Befehl lädt alle Dateien im Unterverzeichnis „Scripts“ in den Bucket „test-files“ hoch und wendet das gemeinsame key prefix "SampleScripts" auf jedes Objekt an. Jede hochgeladene Datei hat den Schlüssel "SampleScripts/filename", wobei 'Dateiname' variiert.

```
Write-S3Object -BucketName test-files -Folder .\Scripts -KeyPrefix SampleScripts\
```

Beispiel 5: Dieser Befehl lädt alle*.ps1-Dateien im lokalen Verzeichnis „Scripts“ in den Bucket „test-files“ hoch und wendet das gemeinsame key prefix "" SampleScripts auf jedes Objekt an. Jede hochgeladene Datei hat den Schlüssel "SampleScripts/filename.ps1", wobei 'Dateiname' variiert.

```
Write-S3Object -BucketName test-files -Folder .\Scripts -KeyPrefix SampleScripts\ -
SearchPattern *.ps1
```

Beispiel 6: Dieser Befehl erstellt ein neues S3-Objekt, das die angegebene Inhaltszeichenfolge mit dem Schlüssel 'sample.txt' enthält.

```
Write-S3Object -BucketName test-files -Key "sample.txt" -Content "object contents"
```

Beispiel 7: Dieser Befehl lädt die angegebene Datei hoch (der Dateiname wird als Schlüssel verwendet) und wendet die angegebenen Tags auf das neue Objekt an.

```
Write-S3Object -BucketName test-files -File "sample.txt" -TagSet
 @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

Beispiel 8: Dieser Befehl lädt den angegebenen Ordner rekursiv hoch und wendet die angegebenen Tags auf alle neuen Objekte an.

```
Write-S3Object -BucketName test-files -Folder . -KeyPrefix "TaggedFiles" -Recurse -
TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- Einzelheiten zur API finden Sie unter PutObject AWS Tools for PowerShellCmdlet-Referenz.

`Write-S3ObjectRetention`

Das folgende Codebeispiel zeigt die Verwendung. `Write-S3ObjectRetention`

Tools für PowerShell

Beispiel 1: Der Befehl aktiviert den Governance-Aufbewahrungsmodus bis zum Datum „31. Dezember 2019 00:00:00“ für das Objekt 'testfile.txt' im angegebenen S3-Bucket.

```
Write-S3ObjectRetention -BucketName 's3buckettesting' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- Einzelheiten zur API finden Sie unter PutObjectRetentionCmdlet-Referenz.AWS Tools for PowerShell

S3 Glacier-Beispiele mit Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von S3 Glacier Aktionen ausführen und allgemeine Szenarien implementieren. AWS Tools for PowerShell

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

Aktionen

`Get-GLCJob`

Das folgende Codebeispiel zeigt die Verwendung`Get-GLCJob`.

Tools für PowerShell

Beispiel 1: Gibt Details des angegebenen Jobs zurück. Wenn der Auftrag erfolgreich abgeschlossen wurde, kann das JobOutput Cmdlet Read-GC verwendet werden, um den Inhalt des Auftrags (ein Archiv oder eine Inventarliste) in das lokale Dateisystem abzurufen.

```
Get-GLCJob -VaultName myvault -JobId "op1x...JSbthM"
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeJobCmdlet-Referenz.AWS Tools for PowerShell

`New-GLCVault`

Das folgende Codebeispiel zeigt die Verwendung. `New-GLCVault`

Tools für PowerShell

Beispiel 1: Erstellt einen neuen Tresor für das Konto des Benutzers. Da für den AccountId Parameter - kein Wert angegeben wurde, verwenden die Cmdlets den Standardwert „-“, der das aktuelle Konto angibt.

```
New-GLCVault -VaultName myvault
```

Ausgabe:

```
/01234567812/vaults/myvault
```

- Einzelheiten zur API finden Sie unter CreateVault AWS Tools for PowerShellCmdlet-Referenz.

`Read-GLCJobOutput`

Das folgende Codebeispiel zeigt die Verwendung. `Read-GLCJobOutput`

Tools für PowerShell

Beispiel 1: Lädt den Archivinhalt herunter, dessen Abruf für den angegebenen Job geplant war, und speichert den Inhalt in einer Datei auf der Festplatte. Der Download validiert die Prüfsumme für Sie, sofern eine verfügbar ist. Falls erforderlich, kann die Prüfsumme wie folgt aus dem Antwortverlauf des Dienstes abgerufen werden (vorausgesetzt, dieses Cmdlet wurde zuletzt

ausgeführt):. **`$AWSHistory.LastServiceResponse`** Wenn das Cmdlet nicht zuletzt ausgeführt wurde, überprüfen Sie die **`$AWSHistory.Commands`** Sammlung, um die entsprechende Dienstantwort zu erhalten.

```
Read-GLCJobOutput -VaultName myvault -JobId "HSWjArc...Zq2XLiW" -FilePath "c:\temp
\blue.bin"
```

- Einzelheiten zur API finden Sie unter GetJobOutput AWS Tools for PowerShellCmdlet-Referenz.

`Start-GLCJob`

Das folgende Codebeispiel zeigt die Verwendung. `Start-GLCJob`

Tools für PowerShell

Beispiel 1: Startet einen Job zum Abrufen eines Archivs aus dem angegebenen Tresor, der dem Benutzer gehört. Der Status des Auftrags kann mit dem Cmdlet Get-GlcJob überprüft werden. Wenn der Auftrag erfolgreich abgeschlossen wurde, kann das JobOutput Cmdlet Read-GC verwendet werden, um den Inhalt des Archivs in das lokale Dateisystem abzurufen.

```
Start-GLCJob -VaultName myvault -JobType "archive-retrieval" -JobDescription
 "archive retrieval" -ArchiveId "o9O9j...TX-TpIhQJw"
```

Ausgabe:

```
JobId            JobOutputPath Location
-----            ------------- --------
op1x...JSbthM                  /012345678912/vaults/test/jobs/op1xe...I4HqCHkSJSbthM
```

- Einzelheiten zur API finden Sie unter InitiateJobCmdlet-Referenz.AWS Tools for PowerShell

`Write-GLCArchive`

Das folgende Codebeispiel zeigt die Verwendung. `Write-GLCArchive`

Tools für PowerShell

Beispiel 1: Lädt eine einzelne Datei in den angegebenen Tresor hoch und gibt die Archiv-ID und die berechnete Prüfsumme zurück.

```
Write-GLCArchive -VaultName myvault -FilePath c:\temp\blue.bin
```

Ausgabe:

```
FilePath                  ArchiveId             Checksum
--------                  ---------             --------
C:\temp\blue.bin          o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
```

Beispiel 2: Lädt den Inhalt einer Ordnerhierarchie in den angegebenen Tresor im Benutzerkonto hoch. Für jede hochgeladene Datei gibt das Cmdlet den Dateinamen, die entsprechende Archiv-ID und die berechnete Prüfsumme des Archivs aus.

```
Write-GLCArchive -VaultName myvault -FolderPath . -Recurse
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz. UploadArchiveAWS Tools for PowerShell

Amazon SES SES-Beispiele für die Verwendung von Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon SES Aktionen ausführen und allgemeine Szenarien implementieren. AWS Tools for PowerShell

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zur Einrichtung und Ausführung des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`Get-SESIdentity`

Das folgende Codebeispiel zeigt die Verwendung`Get-SESIdentity`.

Tools für PowerShell

Beispiel 1: Dieser Befehl gibt eine Liste zurück, die alle Identitäten (E-Mail-Adressen und Domains) für ein bestimmtes AWS Konto enthält, unabhängig vom Bestätigungsstatus.

```
Get-SESIdentity
```

- Einzelheiten zur API finden Sie unter ListIdentities AWS Tools for PowerShellCmdlet-Referenz.

`Get-SESSendQuota`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SESSendQuota`

Tools für PowerShell

Beispiel 1: Dieser Befehl gibt die aktuellen Sendelimits des Benutzers zurück.

```
Get-SESSendQuota
```

- Einzelheiten zur API finden Sie unter GetSendQuota AWS Tools for PowerShellCmdlet-Referenz.

`Get-SESSendStatistic`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SESSendStatistic`

Tools für PowerShell

Beispiel 1: Dieser Befehl gibt die Sendestatistiken des Benutzers zurück. Das Ergebnis ist eine Liste von Datenpunkten, die die Sendeaktivitäten der letzten zwei Wochen repräsentieren. Jeder Datenpunkt in der Liste enthält Statistiken für ein 15-Minuten-Intervall.

```
Get-SESSendStatistic
```

- Einzelheiten zur API finden Sie unter GetSendStatistics AWS Tools for PowerShellCmdlet-Referenz.

Amazon SNS SNS-Beispiele für die Verwendung von Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS Tools for PowerShell mit Amazon SNS Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`Publish-SNSMessage`

Das folgende Codebeispiel zeigt die Verwendung`Publish-SNSMessage`.

Tools für PowerShell

Beispiel 1: Dieses Beispiel zeigt das Veröffentlichen einer Nachricht mit einer einzigen MessageAttribute deklarierten Inline.

```
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -Message
 "Hello" -MessageAttribute
 @{'City'=[Amazon.SimpleNotificationService.Model.MessageAttributeValue]@{DataType='String';
 StringValue ='AnyCity'}}
```

Beispiel 2: Dieses Beispiel zeigt die Veröffentlichung einer Nachricht, bei der mehrere Nachrichten im Voraus MessageAttributes deklariert wurden.

```
$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
```

- Einzelheiten zur API finden Sie unter In AWS Tools for PowerShell Cmdlet-Referenz veröffentlichen.

Amazon SQS SQS-Beispiele mit Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS Tools for PowerShell mit Amazon SQS Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zur Einrichtung und Ausführung des Codes im Kontext finden.

Themen

Aktionen

`Add-SQSPermission`

Das folgende Codebeispiel zeigt die Verwendung`Add-SQSPermission`.

Tools für PowerShell

Beispiel 1: In diesem Beispiel können die angegebenen AWS-Konto Benutzer Nachrichten aus der angegebenen Warteschlange senden.

```
Add-SQSPermission -Action SendMessage -AWSAccountId 80398EXAMPLE -Label
 SendMessagesFromMyQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue
```

- Einzelheiten zur API finden Sie unter AddPermission AWS Tools for PowerShellCmdlet-Referenz.

`Clear-SQSQueue`

Das folgende Codebeispiel zeigt die Verwendung. `Clear-SQSQueue`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden alle Nachrichten aus der angegebenen Warteschlange gelöscht.

```
Clear-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Einzelheiten zur API finden Sie unter PurgeQueue AWS Tools for PowerShellCmdlet-Referenz.

`Edit-SQSMessageVisibility`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-SQSMessageVisibility`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Sichtbarkeits-Timeout für die Nachricht mit dem angegebenen Empfangs-Handle in der angegebenen Warteschlange auf 10 Stunden (10 Stunden x 60 Minuten x 60 Sekunden = 36000 Sekunden) geändert.

```
Edit-SQSMessageVisibility -QueueUrl https://sqs.us-east-1.amazonaws.com/8039EXAMPLE/
MyQueue -ReceiptHandle AQEBgGDh...J/Iqww== -VisibilityTimeout 36000
```

- Einzelheiten zur API finden Sie unter ChangeMessageVisibility AWS Tools for PowerShellCmdlet-Referenz.

`Edit-SQSMessageVisibilityBatch`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-SQSMessageVisibilityBatch`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Sichtbarkeits-Timeout für 2 Nachrichten mit den angegebenen Empfangs-Handles in der angegebenen Warteschlange geändert. Das Sichtbarkeits-Timeout der ersten Nachricht wird auf 10 Stunden geändert (10 Stunden x 60 Minuten x 60 Sekunden = 36000 Sekunden). Das Sichtbarkeits-Timeout der zweiten Nachricht wird auf 5 Stunden geändert (5 Stunden x 60 Minuten x 60 Sekunden = 18000 Sekunden).

```
$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
```

Ausgabe:

```
Failed    Successful

------    ----------

{}        {Request2, Request1}
```

- Einzelheiten zur API finden Sie unter ChangeMessageVisibilityBatch AWS Tools for PowerShellCmdlet-Referenz.

`Get-SQSDeadLetterSourceQueue`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SQSDeadLetterSourceQueue`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die URLs aller Warteschlangen aufgeführt, die auf die angegebene Warteschlange als Warteschlange für unzustellbare Nachrichten angewiesen sind.

```
Get-SQSDeadLetterSourceQueue -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

Ausgabe:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
```

- Einzelheiten zur API finden Sie unter ListDeadLetterSourceQueues AWS Tools for PowerShellCmdlet-Referenz.

`Get-SQSQueue`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SQSQueue`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden alle Warteschlangen aufgelistet.

```
Get-SQSQueue
```

Ausgabe:

```
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
```

Beispiel 2: In diesem Beispiel werden alle Warteschlangen aufgeführt, die mit dem angegebenen Namen beginnen.

```
Get-SQSQueue -QueueNamePrefix My
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter ListQueues AWS Tools for PowerShellCmdlet-Referenz.

`Get-SQSQueueAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SQSQueueAttribute`

Tools für PowerShell

Beispiel 1: Dieses Beispiel listet alle Attribute für die angegebene Warteschlange auf.

```
Get-SQSQueueAttribute -AttributeName All -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Ausgabe:

```
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]...}
```

Beispiel 2: In diesem Beispiel werden nur die angegebenen Attribute für die angegebene Warteschlange separat aufgeführt.

```
Get-SQSQueueAttribute -AttributeName MaximumMessageSize, VisibilityTimeout -QueueUrl
 https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Ausgabe:

```
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]}
```

- Einzelheiten zur API finden Sie unter GetQueueAttributes AWS Tools for PowerShellCmdlet-Referenz.

`Get-SQSQueueUrl`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SQSQueueUrl`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die URL der Warteschlange mit dem angegebenen Namen aufgeführt.

```
Get-SQSQueueUrl -QueueName MyQueue
```

Ausgabe:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Einzelheiten zur API finden Sie unter GetQueueUrl AWS Tools for PowerShellCmdlet-Referenz.

`New-SQSQueue`

Das folgende Codebeispiel zeigt die Verwendung. `New-SQSQueue`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Warteschlange mit dem angegebenen Namen erstellt.

```
New-SQSQueue -QueueName MyQueue
```

Ausgabe:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Einzelheiten zur API finden Sie unter CreateQueue AWS Tools for PowerShellCmdlet-Referenz.

`Receive-SQSMessage`

Das folgende Codebeispiel zeigt die Verwendung. `Receive-SQSMessage`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Informationen für die nächsten 10 Nachrichten aufgelistet, die in der angegebenen Warteschlange empfangen werden sollen. Die Informationen enthalten Werte für die angegebenen Nachrichtenattribute, sofern sie existieren.

```
Receive-SQSMessage -AttributeName SenderId, SentTimestamp -MessageAttributeName
 StudentName, StudentGrade -MessageCount 10 -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Ausgabe:

```
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==
```

- Einzelheiten zur API finden Sie unter ReceiveMessage AWS Tools for PowerShellCmdlet-Referenz.

`Remove-SQSMessage`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-SQSMessage`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Nachricht mit dem angegebenen Empfangs-Handle aus der angegebenen Warteschlange gelöscht.

```
Remove-SQSMessage -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
 -ReceiptHandle AQEBd329...v6gl8Q==
```

- Einzelheiten zur API finden Sie unter DeleteMessage AWS Tools for PowerShellCmdlet-Referenz.

`Remove-SQSMessageBatch`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-SQSMessageBatch`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden 2 Nachrichten mit den angegebenen Empfangs-Handles aus der angegebenen Warteschlange gelöscht.

```
$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
```

Ausgabe:

```
Failed    Successful

------    ----------

{}        {Request1, Request2}
```

- Einzelheiten zur API finden Sie unter DeleteMessageBatch AWS Tools for PowerShellCmdlet-Referenz.

`Remove-SQSPermission`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-SQSPermission`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Berechtigungseinstellungen mit dem angegebenen Label aus der angegebenen Warteschlange entfernt.

```
Remove-SQSPermission -Label SendMessagesFromMyQueue -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Einzelheiten zur API finden Sie unter RemovePermission AWS Tools for PowerShellCmdlet-Referenz.

`Remove-SQSQueue`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-SQSQueue`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Warteschlange gelöscht.

```
Remove-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Einzelheiten zur API finden Sie unter DeleteQueue AWS Tools for PowerShellCmdlet-Referenz.

`Send-SQSMessage`

Das folgende Codebeispiel zeigt die Verwendung. `Send-SQSMessage`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Nachricht mit den angegebenen Attributen und dem Nachrichtentext an die angegebene Warteschlange gesendet, wobei die Nachrichtenzustellung um 10 Sekunden verzögert wird.

```
$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
```

Ausgabe:

```
MD5OfMessageAttributes             MD5OfMessageBody                    MessageId

----------------------             ----------------                    ---------

1d3e51347bc042efbdf6dda31EXAMPLE   51b0a3256d59467f973009b73EXAMPLE    c35fed8f-
c739-4d0c-818b-1820eEXAMPLE
```

- Einzelheiten zur API finden Sie unter SendMessage AWS Tools for PowerShellCmdlet-Referenz.

`Send-SQSMessageBatch`

Das folgende Codebeispiel zeigt die Verwendung. `Send-SQSMessageBatch`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden 2 Nachrichten mit den angegebenen Attributen und Nachrichtentexten an die angegebene Warteschlange gesendet. Die Zustellung verzögert sich bei der ersten Nachricht um 15 Sekunden und bei der zweiten Nachricht um 10 Sekunden.

```
$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
```

Ausgabe:

```
Failed    Successful

------    ----------

{}        {FirstMessage, SecondMessage}
```

- Einzelheiten zur API finden Sie unter SendMessageBatch AWS Tools for PowerShellCmdlet-Referenz.

`Set-SQSQueueAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Set-SQSQueueAttribute`

Beispiel 1: Dieses Beispiel zeigt, wie eine Richtlinie eingerichtet wird, die eine Warteschlange für ein SNS-Thema abonniert. Wenn eine Nachricht zu dem Thema veröffentlicht wird, wird eine Nachricht an die abonnierte Warteschlange gesendet.

```
# 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 }
```

Beispiel 2: In diesem Beispiel werden die angegebenen Attribute für die angegebene Warteschlange festgelegt.

```
Set-SQSQueueAttribute -Attribute @{"DelaySeconds" = "10"; "MaximumMessageSize" =
 "131072"} -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Einzelheiten zur API finden Sie unter SetQueueAttributes AWS Tools for PowerShellCmdlet-Referenz.

AWS STS Beispiele für die Verwendung von Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS Tools for PowerShell with Aktionen ausführen und allgemeine Szenarien implementieren AWS STS.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`Convert-STSAuthorizationMessage`

Das folgende Codebeispiel zeigt die Verwendung`Convert-STSAuthorizationMessage`.

Tools für PowerShell

Beispiel 1: Dekodiert die zusätzlichen Informationen, die im bereitgestellten codierten Nachrichteninhalt enthalten sind und als Antwort auf eine Anfrage zurückgegeben wurden. Die zusätzlichen Informationen sind verschlüsselt, da es sich bei Details zum Autorisierungsstatus um vertrauliche Informationen handeln kann, die der Benutzer, der die Aktion angefordert hat, nicht sehen sollte.

```
Convert-STSAuthorizationMessage -EncodedMessage "...encoded message..."
```

- Einzelheiten zur API finden Sie unter DecodeAuthorizationMessage AWS Tools for PowerShellCmdlet-Referenz.

`Get-STSFederationToken`

Das folgende Codebeispiel zeigt die Verwendung. `Get-STSFederationToken`

Tools für PowerShell

Beispiel 1: Fordert ein Verbundtoken an, das für eine Stunde gültig ist, wobei „Bob“ als Name des Verbundbenutzers verwendet wird. Dieser Name kann verwendet werden, um in einer ressourcenbasierten Richtlinie (z. B. einer Amazon S3 S3-Bucket-Richtlinie) auf den Verbundbenutzernamen zu verweisen. Die bereitgestellte IAM-Richtlinie im JSON-Format wird verwendet, um den Umfang der Berechtigungen einzuschränken, die dem IAM-Benutzer zur Verfügung stehen. Die bereitgestellte Richtlinie kann nicht mehr Berechtigungen gewähren als die, die dem anfragenden Benutzer gewährt werden, wobei die endgültigen Berechtigungen für den Verbundbenutzer aufgrund der Schnittmenge zwischen der übergebenen Richtlinie und der IAM-Benutzerrichtlinie am restriktivsten sind.

```
Get-STSFederationToken -Name "Bob" -Policy "...JSON policy..." -DurationInSeconds
 3600
```

- Einzelheiten zur API finden Sie unter GetFederationToken AWS Tools for PowerShellCmdlet-Referenz.

`Get-STSSessionToken`

Das folgende Codebeispiel zeigt die Verwendung. `Get-STSSessionToken`

Tools für PowerShell

Beispiel 1: Gibt eine **`Amazon.RuntimeAWSCredentials`** Instanz zurück, die temporäre Anmeldeinformationen enthält, die für einen bestimmten Zeitraum gültig sind. Die Anmeldeinformationen, die zum Anfordern temporärer Anmeldeinformationen verwendet werden, werden aus den aktuellen Shell-Standardeinstellungen abgeleitet. Um andere Anmeldeinformationen anzugeben, verwenden Sie die Parameter - ProfileName oder - AccessKey SecretKey /-.

```
Get-STSSessionToken
```

Ausgabe:

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

Beispiel 2: Gibt eine **`Amazon.RuntimeAWSCredentials`** Instanz zurück, die temporäre Anmeldeinformationen enthält, die für eine Stunde gültig sind. Die für die Anfrage verwendeten Anmeldeinformationen stammen aus dem angegebenen Profil.

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile
```

Ausgabe:

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

Beispiel 3: Gibt eine **`Amazon.RuntimeAWSCredentials`** Instanz mit temporären Anmeldeinformationen zurück, die für eine Stunde gültig sind. Dabei werden die Identifikationsnummer des MFA-Geräts, das dem Konto zugeordnet ist, dessen Anmeldeinformationen im Profil „myprofilename“ angegeben sind, und der vom Gerät bereitgestellte Wert verwendet.

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile -SerialNumber
 YourMFADeviceSerialNumber -TokenCode 123456
```

Ausgabe:

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
```

```
EXAMPLEACCESSKEYID                    2/16/2015 9:12:28 PM
 examplesecretaccesskey...            SamPleTokeN.....
```

- Einzelheiten zur API finden Sie unter GetSessionTokenCmdlet-Referenz.AWS Tools for PowerShell

`Use-STSRole`

Das folgende Codebeispiel zeigt die Verwendung. `Use-STSRole`

Tools für PowerShell

Beispiel 1: Gibt einen Satz temporärer Anmeldeinformationen (Zugriffsschlüssel, geheimer Schlüssel und Sitzungstoken) zurück, die eine Stunde lang für den Zugriff auf AWS Ressourcen verwendet werden können, auf die der anfragende Benutzer normalerweise keinen Zugriff hat. Die zurückgegebenen Anmeldeinformationen verfügen über die Berechtigungen, die durch die Zugriffsrichtlinie der übernommenen Rolle und die angegebene Richtlinie zulässig sind (Sie können die bereitgestellte Richtlinie nicht verwenden, um Berechtigungen zu gewähren, die über die in der Zugriffsrichtlinie der übernommenen Rolle definierten Berechtigungen hinausgehen).

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
Policy "...JSON policy..." -DurationInSeconds 3600
```

Beispiel 2: Gibt einen Satz temporärer Anmeldeinformationen zurück, die für eine Stunde gültig sind und dieselben Berechtigungen haben, die in der Zugriffsrichtlinie der Rolle definiert sind, die angenommen wird.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
DurationInSeconds 3600
```

Beispiel 3: Gibt einen Satz temporärer Anmeldeinformationen zurück, die die Seriennummer und das generierte Token aus einem MFA enthalten, das den Benutzeranmeldedaten zugeordnet ist, die zur Ausführung des Cmdlets verwendet wurden.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
DurationInSeconds 3600 -SerialNumber "GAHT12345678" -TokenCode "123456"
```

Beispiel 4: Gibt einen Satz temporärer Anmeldeinformationen zurück, die eine in einem Kundenkonto definierte Rolle übernommen haben. Für jede Rolle, die der Drittanbieter

übernehmen kann, muss das Kundenkonto eine Rolle mithilfe einer Kennung erstellen, die bei jeder Übernahme der Rolle im ExternalId Parameter - übergeben werden muss.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
DurationInSeconds 3600 -ExternalId "ABC123"
```

- Einzelheiten zur API finden Sie unter AssumeRole AWS Tools for PowerShellCmdlet-Referenz.

`Use-STSWebIdentityRole`

Das folgende Codebeispiel zeigt die Verwendung. `Use-STSWebIdentityRole`

Tools für PowerShell

Beispiel 1: Gibt einen temporären Satz von Anmeldeinformationen zurück, der für eine Stunde gültig ist, für einen Benutzer, der mit dem Identity-Provider „Login with Amazon“ authentifiziert wurde. Die Anmeldeinformationen gehen von der Zugriffsrichtlinie aus, die mit der Rolle verknüpft ist, die durch den Rollen-ARN identifiziert wurde. Optional können Sie dem Parameter -Policy eine JSON-Richtlinie übergeben, die die Zugriffsberechtigungen weiter verfeinert (Sie können nicht mehr Berechtigungen gewähren, als in den mit der Rolle verknüpften Berechtigungen verfügbar sind). Der an - übergebene Wert WebIdentityToken ist die eindeutige Benutzer-ID, die vom Identitätsanbieter zurückgegeben wurde.

```
Use-STSWebIdentityRole -DurationInSeconds 3600 -ProviderId "www.amazon.com"
 -RoleSessionName "app1" -RoleArn "arn:aws:iam::123456789012:role/
FederatedWebIdentityRole" -WebIdentityToken "Atza...DVI0r1"
```

- Einzelheiten zur API finden Sie unter AssumeRoleWithWebIdentity AWS Tools for PowerShellCmdlet-Referenz.

AWS Support Beispiele für die Verwendung von Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS Tools for PowerShell with Aktionen ausführen und allgemeine Szenarien implementieren AWS Support.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

Aktionen

`Add-ASACommunicationToCase`

Das folgende Codebeispiel zeigt die Verwendung`Add-ASACommunicationToCase`.

Tools für PowerShell

Beispiel 1: Fügt dem angegebenen Fall den Text einer E-Mail-Kommunikation hinzu.

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -
CommunicationBody "Some text about the case"
```

Beispiel 2: Fügt dem angegebenen Fall den Text einer E-Mail-Nachricht sowie eine oder mehrere E-Mail-Adressen hinzu, die in der CC-Zeile der E-Mail enthalten sind.

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -
CcEmailAddress @("email1@address.com", "email2@address.com") -CommunicationBody
 "Some text about the case"
```

- Einzelheiten zur API finden Sie unter AddCommunicationToCase AWS Tools for PowerShellCmdlet-Referenz.

`Get-ASACase`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASACase`

Tools für PowerShell

Beispiel 1: Gibt die Details aller Supportfälle zurück.

```
Get-ASACase
```

Beispiel 2: Gibt die Details aller Supportfälle seit dem angegebenen Datum und der angegebenen Uhrzeit zurück.

```
Get-ASACase -AfterTime "2013-09-10T03:06Z"
```

Beispiel 3: Gibt die Details der ersten 10 Supportfälle zurück, einschließlich derer, die gelöst wurden.

```
Get-ASACase -MaxResult 10 -IncludeResolvedCases $true
```

Beispiel 4: Gibt die Details des einzelnen angegebenen Supportfalls zurück.

```
Get-ASACase -CaseIdList "case-12345678910-2013-c4c1d2bf33c5cf47"
```

Beispiel 5: Gibt die Details der angegebenen Supportfälle zurück.

```
Get-ASACase -CaseIdList @("case-12345678910-2013-c4c1d2bf33c5cf47",
 "case-18929034710-2011-c4fdeabf33c5cf47")
```

Beispiel 6: Gibt alle Supportanfragen mithilfe von manuellem Paging zurück. Die Anfragen werden in Stapeln von 20 abgerufen.

```
$nextToken = $null
do {
  Get-ASACase -NextToken $nextToken -MaxResult 20
  $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Einzelheiten zur API finden Sie unter DescribeCases AWS Tools for PowerShellCmdlet-Referenz.

`Get-ASACommunication`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASACommunication`

Beispiel 1: Gibt die gesamte Kommunikation für den angegebenen Fall zurück.

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

Beispiel 2: Gibt für den angegebenen Fall alle Mitteilungen seit Mitternacht UTC am 1. Januar 2012 zurück.

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -AfterTime
 "2012-01-10T00:00Z"
```

Beispiel 3: Gibt alle Kommunikationen seit Mitternacht UTC am 1. Januar 2012 für den angegebenen Fall zurück, wobei manuelles Paging verwendet wird. Die Mitteilungen werden in Stapeln von 20 abgerufen.

```
$nextToken = $null
do {
  Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -NextToken
 $nextToken -MaxResult 20
  $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Einzelheiten zur API finden Sie unter DescribeCommunications AWS Tools for PowerShellCmdlet-Referenz.

`Get-ASAService`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASAService`

Tools für PowerShell

Beispiel 1: Gibt alle verfügbaren Servicecodes, Namen und Kategorien zurück.

```
Get-ASAService
```

Beispiel 2: Gibt den Namen und die Kategorien für den Dienst mit dem angegebenen Code zurück.

```
Get-ASAService -ServiceCodeList "amazon-cloudfront"
```

Beispiel 3: Gibt den Namen und die Kategorien für die angegebenen Servicecodes zurück.

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch")
```

Beispiel 4: Gibt den Namen und die Kategorien (auf Japanisch) für die angegebenen Servicecodes zurück. Derzeit werden die Sprachcodes Englisch („en“) und Japanisch („ja“) unterstützt.

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch") -
Language "ja"
```

- Einzelheiten zur API finden Sie unter DescribeServices AWS Tools for PowerShellCmdlet-Referenz.

`Get-ASASeverityLevel`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASASeverityLevel`

Tools für PowerShell

Beispiel 1: Gibt die Liste der Schweregrade zurück, die einem AWS Support-Fall zugewiesen werden können.

```
Get-ASASeverityLevel
```

Beispiel 2: Gibt die Liste der Schweregrade zurück, die einem AWS Support-Fall zugewiesen werden können. Die Namen der Stufen werden auf Japanisch zurückgegeben.

```
Get-ASASeverityLevel -Language "ja"
```

- Einzelheiten zur API finden Sie unter DescribeSeverityLevels AWS Tools for PowerShellCmdlet-Referenz.

`Get-ASATrustedAdvisorCheck`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASATrustedAdvisorCheck`

Tools für PowerShell

Beispiel 1: Gibt die Sammlung von Trusted Advisor Advisor-Checks zurück. Sie müssen den Sprachparameter angeben, der entweder „en“ für die englische Ausgabe oder „ja“ für die japanische Ausgabe akzeptiert.

```
Get-ASATrustedAdvisorCheck -Language "en"
```

- Einzelheiten zur API finden Sie unter DescribeTrustedAdvisorChecks AWS Tools for PowerShellCmdlet-Referenz.

`Get-ASATrustedAdvisorCheckRefreshStatus`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASATrustedAdvisorCheckRefreshStatus`

Tools für PowerShell

Beispiel 1: Gibt den aktuellen Status der Aktualisierungsanforderungen für die angegebenen Prüfungen zurück. Request-ASA TrustedAdvisorCheckRefresh kann verwendet werden, um anzufordern, dass die Statusinformationen der Prüfungen aktualisiert werden.

```
Get-ASATrustedAdvisorCheckRefreshStatus -CheckId @("checkid1", "checkid2")
```

- Einzelheiten zur API finden Sie unter DescribeTrustedAdvisorCheckRefreshStatusesCmdlet-Referenz.AWS Tools for PowerShell

`Get-ASATrustedAdvisorCheckResult`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASATrustedAdvisorCheckResult`

Tools für PowerShell

Beispiel 1: Gibt die Ergebnisse einer Trusted Advisor zurück. Die Liste der verfügbaren Trusted Advisor Advisor-Prüfungen kann mit Get-ASA abgerufen TrustedAdvisorChecks werden. Die Ausgabe enthält den Gesamtstatus der Prüfung, den Zeitstempel, zu dem die Prüfung zuletzt ausgeführt wurde, und die eindeutige Prüf-ID für die jeweilige Prüfung. Um die Ergebnisse auf Japanisch ausgeben zu lassen, fügen Sie den Parameter -Language „ja“ hinzu.

```
Get-ASATrustedAdvisorCheckResult -CheckId "checkid1"
```

- Einzelheiten zur API finden Sie unter DescribeTrustedAdvisorCheckResult AWS Tools for PowerShellCmdlet-Referenz.

`Get-ASATrustedAdvisorCheckSummary`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASATrustedAdvisorCheckSummary`

Tools für PowerShell

Beispiel 1: Gibt die neueste Zusammenfassung für die angegebene Trusted Advisor Advisor-Prüfung zurück.

```
Get-ASATrustedAdvisorCheckSummary -CheckId "checkid1"
```

Beispiel 2: Gibt die neuesten Zusammenfassungen für die angegebenen Trusted Advisor Advisor-Prüfungen zurück.

```
Get-ASATrustedAdvisorCheckSummary -CheckId @("checkid1", "checkid2")
```

- Einzelheiten zur API finden Sie unter DescribeTrustedAdvisorCheckSummaries AWS Tools for PowerShellCmdlet-Referenz.

`New-ASACase`

Das folgende Codebeispiel zeigt die Verwendung. `New-ASACase`

Tools für PowerShell

Beispiel 1: Erstellt einen neuen Fall im AWS Support Center. Werte für die CategoryCode Parameter - ServiceCode und - können mit dem Cmdlet Get-AsaService abgerufen werden. Der Wert für den SeverityCode Parameter - kann mit dem Cmdlet Get-ASA abgerufen werden. SeverityLevel Der Wert des IssueType Parameters - kann entweder „Kundenservice“ oder „technisch“ sein. Bei Erfolg wird die AWS Support-Fallnummer ausgegeben. Standardmäßig wird der Fall auf Englisch behandelt. Um Japanisch zu verwenden, fügen Sie den Parameter - Language „ja“ hinzu. Die CommunicationBody Parameter -ServiceCode, -CategoryCode, -Subject und - sind verpflichtend.

```
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"
```

- Einzelheiten zur API finden Sie unter CreateCase AWS Tools for PowerShellCmdlet-Referenz.

`Request-ASATrustedAdvisorCheckRefresh`

Das folgende Codebeispiel zeigt die Verwendung. `Request-ASATrustedAdvisorCheckRefresh`

Tools für PowerShell

Beispiel 1: Fordert eine Aktualisierung für die angegebene Trusted Advisor Advisor-Prüfung an.

```
Request-ASATrustedAdvisorCheckRefresh -CheckId "checkid1"
```

- Einzelheiten zur API finden Sie unter RefreshTrustedAdvisorCheck AWS Tools for PowerShellCmdlet-Referenz.

`Resolve-ASACase`

Das folgende Codebeispiel zeigt die Verwendung. `Resolve-ASACase`

Tools für PowerShell

Beispiel 1: Gibt den Anfangsstatus des angegebenen Falls und den aktuellen Status nach Abschluss des Aufrufs zur Lösung des Falls zurück.

```
Resolve-ASACase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

- Einzelheiten zur API finden Sie unter ResolveCase AWS Tools for PowerShellCmdlet-Referenz.

Systems Manager Manager-Beispiele mit Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Systems Manager Aktionen ausführen und allgemeine Szenarien implementieren. AWS Tools for PowerShell

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

Aktionen

`Add-SSMResourceTag`

Das folgende Codebeispiel zeigt die Verwendung`Add-SSMResourceTag`.

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Wartungsfenster mit neuen Tags aktualisiert. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$option1 = @{Key="Stack";Value=@("Production")}
Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -Tag $option1
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um jedes Tag zu erstellen. Es erfolgt keine Ausgabe, wenn der Befehl erfolgreich ist.

```
$tag1 = New-Object Amazon.SimpleSystemsManagement.Model.Tag
$tag1.Key = "Stack"
$tag1.Value = "Production"

Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -Tag $tag1
```

- Einzelheiten zur API finden Sie unter AddTagsToResource AWS Tools for PowerShellCmdlet-Referenz.

`Edit-SSMDocumentPermission`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-SSMDocumentPermission`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden allen Konten für ein Dokument „Teilen“ -Berechtigungen hinzugefügt. Es erfolgt keine Ausgabe, wenn der Befehl erfolgreich ist.

```
Edit-SSMDocumentPermission -Name "RunShellScript" -PermissionType "Share" -
AccountIdsToAdd all
```

Beispiel 2: In diesem Beispiel werden einem bestimmten Konto für ein Dokument „Teilen“ -Berechtigungen hinzugefügt. Es erfolgt keine Ausgabe, wenn der Befehl erfolgreich ist.

```
Edit-SSMDocumentPermission -Name "RunShellScriptNew" -PermissionType "Share" -
AccountIdsToAdd "123456789012"
```

- Einzelheiten zur API finden Sie unter ModifyDocumentPermission AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMActivation`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMActivation`

Tools für PowerShell

Beispiel 1: Dieses Beispiel enthält Details zu den Aktivierungen in Ihrem Konto.

```
Get-SSMActivation
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeActivations AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMAssociation`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMAssociation`

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die Assoziation zwischen einer Instanz und einem Dokument.

```
Get-SSMAssociation -InstanceId "i-0000293ffd8c57862" -Name "AWS-UpdateSSMAgent"
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeAssociation AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMAssociationExecution`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMAssociationExecution`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Ausführungen für die angegebene Zuordnungs-ID zurückgegeben

```
Get-SSMAssociationExecution -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeAssociationExecutions AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMAssociationExecutionTarget`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMAssociationExecutionTarget`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Ressourcen-ID und ihr Ausführungsstatus angezeigt, die Teil der Ausführungsziele der Assoziation sind

```
Get-SSMAssociationExecutionTarget -AssociationId 123a45a0-
c678-9012-3456-78901234db5e -ExecutionId 123a45a0-c678-9012-3456-78901234db5e |
 Select-Object ResourceId, Status
```

Ausgabe:

```
ResourceId           Status
----------           ------
i-0b1b2a3456f7a890b  Success
i-01c12a45d6fc7a89f  Success
i-0a1caf234f56d7dc8  Success
i-012a3fd45af6dbcfe  Failed
i-0ddc1df23c4a5fb67  Success
```

Beispiel 2: Dieser Befehl überprüft die jeweilige Ausführung einer bestimmten Automatisierung seit gestern, der ein Befehlsdokument zugeordnet ist. Außerdem wird geprüft, ob die Ausführung der Assoziation fehlgeschlagen ist, und wenn ja, werden die Details zum Befehlsaufruf für die Ausführung zusammen mit der Instanz-ID angezeigt

```
$AssociationExecution= Get-SSMAssociationExecutionTarget -
AssociationId 1c234567-890f-1aca-a234-5a678d901cb0 -ExecutionId
 12345ca12-3456-2345-2b45-23456789012 |
    Where-Object {$_.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
    }
}
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeAssociationExecutionTargets AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMAssociationList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMAssociationList`

Beispiel 1: In diesem Beispiel werden alle Assoziationen für eine Instanz aufgeführt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$filter1 = @{Key="InstanceId";Value=@("i-0000293ffd8c57862")}
Get-SSMAssociationList -AssociationFilterList $filter1
```

Ausgabe:

```
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}
```

Beispiel 2: In diesem Beispiel werden alle Verknüpfungen für ein Konfigurationsdokument aufgeführt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$filter2 = @{Key="Name";Value=@("AWS-UpdateSSMAgent")}
Get-SSMAssociationList -AssociationFilterList $filter2
```

Ausgabe:

```
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}
```

Beispiel 3: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um jeden Filter zu erstellen.

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.AssociationFilter
```

```
$filter1.Key = "InstanceId"
$filter1.Value = "i-0000293ffd8c57862"

Get-SSMAssociationList -AssociationFilterList $filter1
```

Ausgabe:

```
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}
```

- Einzelheiten zur API finden Sie unter ListAssociations AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMAssociationVersionList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMAssociationVersionList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden alle Versionen der bereitgestellten Assoziation abgerufen.

```
Get-SSMAssociationVersionList -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

Ausgabe:

```
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}
```

- Einzelheiten zur API finden Sie unter ListAssociationVersions AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMAutomationExecution`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMAutomationExecution`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Details einer Automatisierungsausführung angezeigt.

```
Get-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"
```

Ausgabe:

```
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...}
```

Beispiel 2: In diesem Beispiel werden die Schrittdetails für die angegebene Automatisierungsausführungs-ID aufgeführt

```
Get-SSMAutomationExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object -ExpandProperty StepExecutions | Select-Object
 StepName, Action, StepStatus, ValidNextSteps
```

Ausgabe:

```
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    {}
```

- Einzelheiten zur API finden Sie unter GetAutomationExecution AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMAutomationExecutionList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMAutomationExecutionList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden alle aktiven und beendeten Automatisierungsausführungen beschrieben, die mit Ihrem Konto verknüpft sind.

```
Get-SSMAutomationExecutionList
```

Ausgabe:

```
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]]}
```

Beispiel 2: In diesem Beispiel werden die Ausführungs-ID, das Dokument und der Start-/Endzeitstempel der Ausführung für Ausführungen angezeigt, bei denen es sich nicht um „Erfolg“ handelt AutomationExecutionStatus

```
Get-SSMAutomationExecutionList | Where-Object AutomationExecutionStatus
 -ne "Success" | Select-Object AutomationExecutionId, DocumentName,
```

```
 AutomationExecutionStatus, ExecutionStartTime, ExecutionEndTime | Format-Table -
AutoSize
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz. DescribeAutomationExecutionsAWS Tools for PowerShell

`Get-SSMAutomationStepExecution`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMAutomationStepExecution`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Informationen über alle aktiven und beendeten Schrittausführungen in einem Automatisierungs-Workflow angezeigt.

```
Get-SSMAutomationStepExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object StepName, Action, StepStatus
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeAutomationStepExecutions AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMAvailablePatch`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMAvailablePatch`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden alle verfügbaren Patches für Windows Server 2012 abgerufen, die den MSRC-Schweregrad Kritisch haben. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$filter1 = @{Key="PRODUCT";Values=@("WindowsServer2012")}
$filter2 = @{Key="MSRC_SEVERITY";Values=@("Critical")}

Get-SSMAvailablePatch -Filter $filter1,$filter2
```

Ausgabe:

```
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
...
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um jeden Filter zu erstellen.

```
$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
```

Beispiel 3: In diesem Beispiel werden alle Updates abgerufen, die in den letzten 20 Tagen veröffentlicht wurden und für Produkte gelten, die 2019 entsprechen WindowsServer

```
Get-SSMAvailablePatch | Where-Object ReleaseDate -ge (Get-Date).AddDays(-20) |
 Where-Object Product -eq "WindowsServer2019" | Select-Object ReleaseDate, Product,
 Title
```

Ausgabe:

```
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)
```

- Einzelheiten zur API finden Sie unter DescribeAvailablePatches AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMCommand`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMCommand`

Beispiel 1: Dieses Beispiel listet alle angeforderten Befehle auf.

```
Get-SSMCommand
```

Ausgabe:

```
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            : {}
```

Beispiel 2: In diesem Beispiel wird der Status eines bestimmten Befehls abgerufen.

```
Get-SSMCommand -CommandId "4b75a163-d39a-4d97-87c9-98ae52c6be35"
```

Beispiel 3: In diesem Beispiel werden alle SSM-Befehle abgerufen, die nach dem 2019-04-01T 00:00:00 Z aufgerufen wurden

```
Get-SSMCommand -Filter @{Key="InvokedAfter";Value="2019-04-01T00:00:00Z"} | Select-
Object CommandId, DocumentName, Status, RequestedDateTime | Sort-Object -Property
 RequestedDateTime -Descending
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz. ListCommandsAWS Tools for PowerShell

`Get-SSMCommandInvocation`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMCommandInvocation`

Tools für PowerShell

Beispiel 1: Dieses Beispiel listet alle Aufrufe eines Befehls auf.

```
Get-SSMCommandInvocation -CommandId "b8eac879-0541-439d-94ec-47a80d554f44" -Detail
 $true
```

Ausgabe:

```
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        :
```

Beispiel 2: In diesem Beispiel wird der Aufruf der Befehls-ID CommandPlugins e1eb2e3c-ed4c-5123-45c1-234f5612345f aufgeführt

```
Get-SSMCommandInvocation -CommandId e1eb2e3c-ed4c-5123-45c1-234f5612345f -Detail:
$true | Select-Object -ExpandProperty CommandPlugins
```

Ausgabe:

```
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
```

- Einzelheiten AWS Tools for PowerShell zur ListCommandInvocationsAPI finden Sie unter Cmdlet-Referenz.

`Get-SSMCommandInvocationDetail`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMCommandInvocationDetail`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Details eines Befehls angezeigt, der auf einer Instanz ausgeführt wurde.

```
Get-SSMCommandInvocationDetail -InstanceId "i-0cb2b964d3e14fd9f" -CommandId
 "b8eac879-0541-439d-94ec-47a80d554f44"
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter GetCommandInvocation AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMComplianceItemList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMComplianceItemList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Liste der Compliance-Elemente für die angegebene Ressourcen-ID und den angegebenen Ressourcentyp aufgeführt, wobei nach dem Compliance-Typ „Association“ gefiltert wird

```
Get-SSMComplianceItemList -ResourceId i-1a2caf345f67d0dc2 -ResourceType
 ManagedInstance -Filter @{Key="ComplianceType";Values="Association"}
```

Ausgabe:

```
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            :
```

- Einzelheiten zur API finden Sie unter ListComplianceItems AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMComplianceSummaryList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMComplianceSummaryList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Zusammenfassung der Anzahl der konformen und nicht konformen Ressourcen für alle Compliance-Typen zurückgegeben.

```
Get-SSMComplianceSummaryList
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter ListComplianceSummaries AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMConnectionStatus`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMConnectionStatus`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der Session Manager-Verbindungsstatus für eine Instanz abgerufen, um festzustellen, ob sie verbunden und bereit ist, Session Manager-Verbindungen zu empfangen.

```
Get-SSMConnectionStatus -Target i-0a1caf234f12d3dc4
```

Ausgabe:

```
Status    Target
------    ------
Connected i-0a1caf234f12d3dc4
```

- Einzelheiten zur API finden Sie unter GetConnectionStatus AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMDefaultPatchBaseline`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMDefaultPatchBaseline`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Standard-Patch-Baseline angezeigt.

```
Get-SSMDefaultPatchBaseline
```

Ausgabe:

```
arn:aws:ssm:us-west-2:123456789012:patchbaseline/pb-04fb4ae6142167966
```

- Einzelheiten zur API finden Sie unter GetDefaultPatchBaseline AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMDeployablePatchSnapshotForInstance`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMDeployablePatchSnapshotForInstance`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der aktuelle Snapshot für die von einer Instance verwendete Patch-Baseline angezeigt. Dieser Befehl muss von der Instance aus mit den Anmeldeinformationen der Instanz ausgeführt werden. Um sicherzustellen, dass die Instanzanmeldedaten verwendet werden, übergibt das Beispiel ein **`Amazon.Runtime.InstanceProfileAWSCredentials`** Objekt an den Credentials-Parameter.

```
$credentials = [Amazon.Runtime.InstanceProfileAWSCredentials]::new()
Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f" -Credentials $credentials
```

Ausgabe:

```
InstanceId          SnapshotDownloadUrl
----------          -------------------
i-0cb2b964d3e14fd9f https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...1692/4681775b-098f-4435...
```

Beispiel 2: Dieses Beispiel zeigt, wie Sie die vollständigen Daten abrufen können SnapshotDownloadUrl. Dieser Befehl muss von der Instanz aus mit den Instanzanmeldedaten ausgeführt werden. Um sicherzustellen, dass die Instanzanmeldedaten verwendet werden, konfiguriert das Beispiel die PowerShell Sitzung für die Verwendung eines **`Amazon.Runtime.InstanceProfileAWSCredentials`** Objekts.

```
Set-AWSCredential -Credential
 ([Amazon.Runtime.InstanceProfileAWSCredentials]::new())
(Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f").SnapshotDownloadUrl
```

Ausgabe:

```
https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...
```

- Einzelheiten zur API finden Sie unter GetDeployablePatchSnapshotForInstance AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMDocument`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMDocument`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der Inhalt eines Dokuments zurückgegeben.

```
Get-SSMDocument -Name "RunShellScript"
```

Ausgabe:

```
Content
-------
{...
```

Beispiel 2: In diesem Beispiel wird der vollständige Inhalt eines Dokuments angezeigt.

```
(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 }}"
         }
      }
   ]
}
```

- Einzelheiten zur API finden Sie unter GetDocument AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMDocumentDescription`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMDocumentDescription`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Informationen zu einem Dokument zurückgegeben.

```
Get-SSMDocumentDescription -Name "RunShellScript"
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeDocument AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMDocumentList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMDocumentList`

Tools für PowerShell

Beispiel 1: Listet alle Konfigurationsdokumente in Ihrem Konto auf.

```
Get-SSMDocumentList
```

Ausgabe:

```
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
...
```

Beispiel 2: In diesem Beispiel werden alle Automatisierungsdokumente abgerufen, deren Name mit „Platform“ übereinstimmt

```
Get-SSMDocumentList -DocumentFilterList @{Key="DocumentType";Value="Automation"} |
 Where-Object Name -Match "Platform"
```

Ausgabe:

```
DocumentFormat  : JSON
DocumentType    : Automation
DocumentVersion : 7
Name            : KT-Get-Platform
Owner           : 987654123456
PlatformTypes   : {Windows, Linux}
SchemaVersion   : 0.3
Tags            : {}
TargetType      :
VersionName     :
```

- Einzelheiten zur API finden Sie unter ListDocumentsCmdlet-Referenz.AWS Tools for PowerShell

`Get-SSMDocumentPermission`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMDocumentPermission`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden alle Versionen eines Dokuments aufgeführt.

```
Get-SSMDocumentVersionList -Name "RunShellScript"
```

Ausgabe:

```
CreatedDate          DocumentVersion IsDefaultVersion Name
-----------          --------------- ---------------- ----
2/24/2017 5:25:13 AM 1               True             RunShellScript
```

- Einzelheiten zur API finden Sie unter DescribeDocumentPermission AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMDocumentVersionList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMDocumentVersionList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Berechtigungsliste für ein Dokument zurückgegeben.

```
Get-SSMDocumentPermission -Name "RunShellScript" -PermissionType "Share"
```

Ausgabe:

```
all
```

- Einzelheiten zur API finden Sie unter ListDocumentVersions AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMEffectiveInstanceAssociationList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMEffectiveInstanceAssociationList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die effektiven Verknüpfungen für eine Instanz beschrieben.

```
Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -MaxResult
 5
```

Ausgabe:

```
AssociationId                        Content
-------------                        -------
d8617c07-2079-4c18-9847-1655fc2698b0 {...
```

Beispiel 2: In diesem Beispiel wird der Inhalt der effektiven Verknüpfungen für eine Instanz angezeigt.

```
(Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -
MaxResult 5).Content
```

Ausgabe:

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

- Einzelheiten zur API finden Sie unter DescribeEffectiveInstanceAssociations AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMEffectivePatchesForPatchBaseline`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMEffectivePatchesForPatchBaseline`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden alle Patch-Baselines mit einer maximalen Ergebnisliste von 1 aufgeführt.

```
Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1
```

Ausgabe:

```
Patch                                       PatchStatus
-----                                       -----------
Amazon.SimpleSystemsManagement.Model.Patch
 Amazon.SimpleSystemsManagement.Model.PatchStatus
```

Beispiel 2: In diesem Beispiel wird der Patchstatus für alle Patch-Baselines mit einer maximalen Ergebnisliste von 1 angezeigt.

```
(Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1).PatchStatus
```

Ausgabe:

```
ApprovalDate          DeploymentStatus
------------          ----------------
12/21/2010 6:00:00 PM APPROVED
```

- Einzelheiten zur API finden Sie unter DescribeEffectivePatchesForPatchBaseline AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMInstanceAssociationsStatus`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMInstanceAssociationsStatus`

Tools für PowerShell

Beispiel 1: Dieses Beispiel zeigt Details der Assoziationen für eine Instanz.

```
Get-SSMInstanceAssociationsStatus -InstanceId "i-0000293ffd8c57862"
```

Ausgabe:

```
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
```

Beispiel 2: In diesem Beispiel wird der Status der Instanzzuweisung für die angegebene Instanz-ID überprüft und außerdem der Ausführungsstatus dieser Zuordnungen angezeigt

```
Get-SSMInstanceAssociationsStatus -InstanceId i-012e3cb4df567e8aa | ForEach-Object
 {Get-SSMAssociationExecution -AssociationId .AssociationId}
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeInstanceAssociationsStatus AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMInstanceInformation`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMInstanceInformation`

Tools für PowerShell

Beispiel 1: Dieses Beispiel zeigt Details zu jeder Ihrer Instanzen.

```
Get-SSMInstanceInformation
```

Ausgabe:

```
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
```

Beispiel 2: Dieses Beispiel zeigt, wie der Parameter -Filter verwendet wird, um Ergebnisse nur nach den AWS Systems Manager Manager-Instanzen in der Region **`us-east-1`** mit dem Wert **`AgentVersion`** von **`2.2.800.0`** zu filtern. Eine Liste der gültigen -Filter-Schlüsselwerte finden Sie im InstanceInformation API-Referenzthema (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
```

Ausgabe:

```
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
```

Beispiel 3: Dieses Beispiel zeigt, wie der InstanceInformationFilterList Parameter - verwendet wird, um Ergebnisse nur nach den AWS Systems Manager Manager-Instanzen in

`PlatformTypes` der Region **`us-east-1`** mit **`Windows`** oder zu filtern**`Linux`**. Eine Liste der gültigen InstanceInformationFilterList Schlüsselwerte finden Sie im InstanceInformationFilter API-Referenzthema (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
```

Ausgabe:

```
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
```

Beispiel 4: In diesem Beispiel werden von SSM verwaltete Instanzen und Exporte InstanceId LastPingDateTime sowie PlatformName in eine CSV-Datei aufgeführt. PingStatus

```
Get-SSMInstanceInformation | Select-Object InstanceId, PingStatus, LastPingDateTime,
 PlatformName | Export-Csv Instance-details.csv -NoTypeInformation
```

- Einzelheiten zur API finden Sie unter DescribeInstanceInformation AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMInstancePatch`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMInstancePatch`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Patch-Compliance-Details für eine Instanz abgerufen.

```
Get-SSMInstancePatch -InstanceId "i-08ee91c0b17045407"
```

- Einzelheiten zur API finden Sie unter DescribeInstancePatches AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMInstancePatchState`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMInstancePatchState`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Status der Patch-Zusammenfassung für eine Instanz abgerufen.

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407"
```

Beispiel 2: In diesem Beispiel werden die Status der Patch-Zusammenfassung für zwei Instanzen abgerufen.

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407","i-09a618aec652973a9"
```

- Einzelheiten zur API finden Sie unter DescribeInstancePatchStates AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMInstancePatchStatesForPatchGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMInstancePatchStatesForPatchGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Status der Patchzusammenfassung pro Instanz für eine Patch-Gruppe abgerufen.

```
Get-SSMInstancePatchStatesForPatchGroup -PatchGroup "Production"
```

- Einzelheiten zur API finden Sie unter DescribeInstancePatchStatesForPatchGroup AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMInventory`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMInventory`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die benutzerdefinierten Metadaten für Ihr Inventar abgerufen.

```
Get-SSMInventory
```

Ausgabe:

```
Data
  Id
----
  --
{[AWS:InstanceInformation,
 Amazon.SimpleSystemsManagement.Model.InventoryResultItem]} i-0cb2b964d3e14fd9f
```

- Einzelheiten zur API finden Sie unter GetInventory AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMInventoryEntriesList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMInventoryEntriesList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden alle benutzerdefinierten Inventareinträge für eine Instanz aufgeführt.

```
Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
 "Custom:RackInfo"
```

Ausgabe:

```
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
```

Beispiel 2: In diesem Beispiel werden die Details aufgeführt.

```
(Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
 "Custom:RackInfo").Entries
```

Ausgabe:

```
Key          Value
---          -----
RackLocation Bay B/Row C/Rack D/Shelf E
```

- Einzelheiten zur API finden Sie unter ListInventoryEntries AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMInventoryEntryList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMInventoryEntryList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden **`AWS:Network`** Typinventareinträge für die Instanz abgerufen.

```
Get-SSMInventoryEntryList -InstanceId mi-088dcb0ecea37b076 -TypeName AWS:Network |
 Select-Object -ExpandProperty Entries
```

Ausgabe:

```
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
```

- API-Details finden Sie unter Get-SSM InventoryEntryList in AWS Tools for PowerShell der Cmdlet-Referenz.

`Get-SSMInventorySchema`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMInventorySchema`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste von Inventartypnamen für das Konto zurückgegeben.

```
Get-SSMInventorySchema
```

- Einzelheiten zur API finden Sie unter GetInventorySchema AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMLatestEC2Image`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMLatestEC2Image`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die neuesten Windows-AMIs aufgeführt.

```
PS Get-SSMLatestEC2Image -Path ami-windows-latest
```

Ausgabe:

```
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
...
```

Beispiel 2: In diesem Beispiel wird die AMI-ID eines bestimmten Amazon Linux-Images für die Region us-west-2 abgerufen.

```
PS Get-SSMLatestEC2Image -Path ami-amazon-linux-latest -ImageName amzn-ami-hvm-
x86_64-ebs -Region us-west-2
```

Ausgabe:

```
ami-09b92cd132204c704
```

Beispiel 3: In diesem Beispiel werden alle neuesten Windows-AMIs aufgeführt, die dem angegebenen Platzhalterausdruck entsprechen.

```
Get-SSMLatestEC2Image -Path ami-windows-latest -ImageName *Windows*2019*English*
```

Ausgabe:

```
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
...
```

- API-Details finden Sie unter Get-SSMLatesTec2Image in der Cmdlet-Referenz.AWS Tools for PowerShell

`Get-SSMMaintenanceWindow`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMMaintenanceWindow`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Details zu einem Wartungsfenster abgerufen.

```
Get-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d"
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter GetMaintenanceWindow AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMMaintenanceWindowExecution`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMMaintenanceWindowExecution`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Informationen über eine Aufgabe aufgeführt, die im Rahmen der Ausführung eines Wartungsfensters ausgeführt wurde.

```
Get-SSMMaintenanceWindowExecution -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter GetMaintenanceWindowExecution AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMMaintenanceWindowExecutionList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMMaintenanceWindowExecutionList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden alle Ausführungen für ein Wartungsfenster aufgeführt.

```
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d"
```

Ausgabe:

```
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
```

Beispiel 2: In diesem Beispiel werden alle Ausführungen für ein Wartungsfenster vor einem bestimmten Datum aufgeführt.

```
$option1 = @{Key="ExecutedBefore";Values=@("2016-11-04T05:00:00Z")}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
 $option1
```

Beispiel 3: In diesem Beispiel werden alle Ausführungen für ein Wartungsfenster nach einem bestimmten Datum aufgeführt.

```
$option1 = @{Key="ExecutedAfter";Values=@("2016-11-04T05:00:00Z")}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
 $option1
```

- Einzelheiten zur API finden Sie unter DescribeMaintenanceWindowExecutions AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMMaintenanceWindowExecutionTask`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMMaintenanceWindowExecutionTask`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Informationen zu einer Aufgabe aufgeführt, die Teil einer Ausführung im Rahmen eines Wartungsfensters war.

```
Get-SSMMaintenanceWindowExecutionTask -TaskId "ac0c6ae1-daa3-4a89-832e-d384503b6586"
 -WindowExecutionId "518d5565-5969-4cca-8f0e-da3b2a638355"
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter GetMaintenanceWindowExecutionTask AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMMaintenanceWindowExecutionTaskInvocationList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMMaintenanceWindowExecutionTaskInvocationList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Aufrufe für eine Aufgabe aufgeführt, die im Rahmen der Ausführung eines Wartungsfensters ausgeführt wurde.

```
Get-SSMMaintenanceWindowExecutionTaskInvocationList -TaskId "ac0c6ae1-
daa3-4a89-832e-d384503b6586" -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"
```

Ausgabe:

```
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    :
```

- Einzelheiten zur API finden Sie unter DescribeMaintenanceWindowExecutionTaskInvocations AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMMaintenanceWindowExecutionTaskList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMMaintenanceWindowExecutionTaskList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Aufgaben aufgeführt, die mit der Ausführung eines Wartungsfensters verbunden sind.

```
Get-SSMMaintenanceWindowExecutionTaskList -WindowExecutionId
 "518d5565-5969-4cca-8f0e-da3b2a638355"
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeMaintenanceWindowExecutionTasks AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMMaintenanceWindowList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMMaintenanceWindowList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden alle Wartungsfenster Ihres Kontos aufgeführt.

```
Get-SSMMaintenanceWindowList
```

Ausgabe:

```
Cutoff   : 1
Duration : 4
Enabled  : True
Name     : My-First-Maintenance-Window
WindowId : mw-06d59c1a07c022145
```

- Einzelheiten zur API finden Sie unter DescribeMaintenanceWindows AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMMaintenanceWindowTarget`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMMaintenanceWindowTarget`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden alle Ziele für ein Wartungsfenster aufgeführt.

```
Get-SSMMaintenanceWindowTarget -WindowId "mw-06cf17cbefcb4bf4f"
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeMaintenanceWindowTargets AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMMaintenanceWindowTaskList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMMaintenanceWindowTaskList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden alle Aufgaben für ein Wartungsfenster aufgeführt.

```
Get-SSMMaintenanceWindowTaskList -WindowId "mw-06cf17cbefcb4bf4f"
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeMaintenanceWindowTasks AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMParameterHistory`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMParameterHistory`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der Werteverlauf für einen Parameter aufgeführt.

```
Get-SSMParameterHistory -Name "Welcome"
```

Ausgabe:

```
Description      :
KeyId            :
```

```
LastModifiedDate : 3/3/2017 6:55:25 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type             : String
Value            : helloWorld
```

- Einzelheiten zur API finden Sie unter GetParameterHistory AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMParameterList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMParameterList`

Tools für PowerShell

Beispiel 1: Dieses Beispiel listet alle Parameter auf.

```
Get-SSMParameterList
```

Ausgabe:

```
Description      :
KeyId            :
LastModifiedDate : 3/3/2017 6:58:23 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type             : String
```

- Einzelheiten zur API finden Sie unter DescribeParameters AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMParameterValue`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMParameterValue`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Werte für einen Parameter aufgeführt.

```
Get-SSMParameterValue -Name "Welcome"
```

Ausgabe:

```
InvalidParameters Parameters
----------------- ----------
{}                {Welcome}
```

Beispiel 2: In diesem Beispiel werden die Details des Werts aufgeführt.

```
(Get-SSMParameterValue -Name "Welcome").Parameters
```

Ausgabe:

```
Name    Type   Value
----    ----   -----
Welcome String Good day, Sunshine!
```

- Einzelheiten zur API finden Sie unter GetParameters AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMPatchBaseline`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMPatchBaseline`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden alle Patch-Baselines aufgeführt.

```
Get-SSMPatchBaseline
```

Ausgabe:

```
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...
```

Beispiel 2: In diesem Beispiel werden alle Patch-Baselines aufgeführt, die von bereitgestellt werden. AWS Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$filter1 = @{Key="OWNER";Values=@("AWS")}
```

Ausgabe:

```
Get-SSMPatchBaseline -Filter $filter1
```

Beispiel 3: In diesem Beispiel werden alle Patch-Baselines mit Ihnen als Eigentümer aufgeführt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$filter1 = @{Key="OWNER";Values=@("Self")}
```

Ausgabe:

```
Get-SSMPatchBaseline -Filter $filter1
```

Beispiel 4: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um jedes Tag zu erstellen.

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "OWNER"
$filter1.Values = "AWS"

Get-SSMPatchBaseline -Filter $filter1
```

Ausgabe:

```
BaselineDescription                     BaselineId
                          BaselineName             DefaultBaselin

                                                   e
-------------------                     ----------
                          ------------             --------------
```

```
Default Patch Baseline Provided by AWS. arn:aws:ssm:us-
west-2:123456789012:patchbaseline/pb-04fb4ae6142167966 AWS-DefaultPatchBaseline True
```

- Einzelheiten zur API finden Sie unter DescribePatchBaselines AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMPatchBaselineDetail`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMPatchBaselineDetail`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Details für eine Patch-Baseline angezeigt.

```
Get-SSMPatchBaselineDetail -BaselineId "pb-03da896ca3b68b639"
```

Ausgabe:

```
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 : {}
```

- Einzelheiten zur API finden Sie unter GetPatchBaseline AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMPatchBaselineForPatchGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMPatchBaselineForPatchGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Patch-Baseline für eine Patch-Gruppe angezeigt.

```
Get-SSMPatchBaselineForPatchGroup -PatchGroup "Production"
```

Ausgabe:

```
BaselineId           PatchGroup
----------           ----------
pb-045f10b4f382baeda Production
```

- Einzelheiten zur API finden Sie unter GetPatchBaselineForPatchGroup AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMPatchGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMPatchGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Registrierungen der Patchgruppen aufgeführt.

```
Get-SSMPatchGroup
```

Ausgabe:

```
BaselineIdentity                                              PatchGroup
----------------                                              ----------
Amazon.SimpleSystemsManagement.Model.PatchBaselineIdentity Production
```

- Einzelheiten zur API finden Sie unter DescribePatchGroups AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMPatchGroupState`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMPatchGroupState`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die allgemeine Zusammenfassung der Patch-Konformität für eine Patch-Gruppe abgerufen.

```
Get-SSMPatchGroupState -PatchGroup "Production"
```

Ausgabe:

```
Instances                          : 4
InstancesWithFailedPatches         : 1
InstancesWithInstalledOtherPatches : 4
InstancesWithInstalledPatches      : 3
InstancesWithMissingPatches        : 0
InstancesWithNotApplicablePatches  : 0
```

- Einzelheiten zur API finden Sie unter DescribePatchGroupState AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMResourceComplianceSummaryList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMResourceComplianceSummaryList`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Zusammenfassung der Anzahl auf Ressourcenebene abgerufen. Die Zusammenfassung enthält Informationen über den Status „konform“ und „nicht konform“ sowie detaillierte Angaben zum Schweregrad von Produkten, die „Windows10" entsprechen. Da der MaxResult Standardwert 100 ist, wenn der Parameter nicht angegeben ist und dieser Wert nicht gültig ist, wird der MaxResult Parameter hinzugefügt und der Wert auf 50 gesetzt.

```
$FilterValues = @{
  "Key"="Product"
        "Type"="EQUAL"
        "Values"="Windows10"
}
        Get-SSMResourceComplianceSummaryList -Filter $FilterValues -MaxResult 50
```

- Einzelheiten zur API finden Sie unter ListResourceComplianceSummaries AWS Tools for PowerShellCmdlet-Referenz.

`Get-SSMResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMResourceTag`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Tags für ein Wartungsfenster aufgelistet.

```
Get-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow"
```

Ausgabe:

```
Key   Value
---   -----
Stack Production
```

- Einzelheiten zur API finden Sie unter ListTagsForResource AWS Tools for PowerShellCmdlet-Referenz.

`New-SSMActivation`

Das folgende Codebeispiel zeigt die Verwendung. `New-SSMActivation`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine verwaltete Instanz erstellt.

```
New-SSMActivation -DefaultInstanceName "MyWebServers" -IamRole "SSMAutomationRole" -
RegistrationLimit 10
```

Ausgabe:

```
ActivationCode       ActivationId
--------------       ------------
KWChhOxBTiwDcKE9BlKC 08e51e79-1e36-446c-8e63-9458569c1363
```

- Einzelheiten zur API finden Sie unter CreateActivation AWS Tools for PowerShellCmdlet-Referenz.

`New-SSMAssociation`

Das folgende Codebeispiel zeigt die Verwendung. `New-SSMAssociation`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Konfigurationsdokument mithilfe von Instanz-IDs einer Instanz zugeordnet.

```
New-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

Ausgabe:

```
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 :
```

Beispiel 2: In diesem Beispiel wird mithilfe von Zielen ein Konfigurationsdokument einer Instanz zugeordnet.

```
$target = @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
New-SSMAssociation -Name "AWS-UpdateSSMAgent" -Target $target
```

Ausgabe:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            :
Date                  : 3/1/2017 6:22:21 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

Beispiel 3: In diesem Beispiel wird ein Konfigurationsdokument mithilfe von Zielen und Parametern einer Instanz zugeordnet.

```
$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
```

Ausgabe:

```
Name                 : Configure-CloudWatch
InstanceId           :
Date                 : 5/17/2018 3:17:44 PM
Status.Name          :
Status.Date          :
Status.Message       :
Status.AdditionalInfo :
```

Beispiel 4: In diesem Beispiel wird eine Assoziation mit allen Instanzen in der Region erstellt, mit**`AWS-GatherSoftwareInventory`**. Außerdem werden benutzerdefinierte Dateien und Registrierungsverzeichnisse in den zu erfassenden Parametern bereitgestellt

```
$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)"
```

Ausgabe:

```
Name                 : AWS-GatherSoftwareInventory
InstanceId           :
Date                 : 6/9/2019 8:57:56 AM
Status.Name          :
Status.Date          :
Status.Message       :
Status.AdditionalInfo :
```

- Einzelheiten zur API finden Sie unter CreateAssociation AWS Tools for PowerShellCmdlet-Referenz.

`New-SSMAssociationFromBatch`

Das folgende Codebeispiel zeigt die Verwendung. `New-SSMAssociationFromBatch`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Konfigurationsdokument mehreren Instanzen zugeordnet. Die Ausgabe gibt gegebenenfalls eine Liste mit erfolgreichen und fehlgeschlagenen Vorgängen zurück.

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@("AWS-UpdateSSMAgent")}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@("AWS-UpdateSSMAgent")}
New-SSMAssociationFromBatch -Entry $option1,$option2
```

Ausgabe:

```
Failed  Successful
------  ----------
{}      {Amazon.SimpleSystemsManagement.Model.FailedCreateAssociation,
 Amazon.SimpleSystemsManagement.Model.FailedCreateAsso...
```

Beispiel 2: In diesem Beispiel werden alle Details eines erfolgreichen Vorgangs angezeigt.

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@("AWS-UpdateSSMAgent")}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@("AWS-UpdateSSMAgent")}
(New-SSMAssociationFromBatch -Entry $option1,$option2).Successful
```

- Einzelheiten zur API finden Sie unter CreateAssociationBatch AWS Tools for PowerShellCmdlet-Referenz.

`New-SSMDocument`

Das folgende Codebeispiel zeigt die Verwendung. `New-SSMDocument`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Dokument in Ihrem Konto erstellt. Das Dokument muss im JSON-Format sein. Weitere Informationen zum Schreiben eines Konfigurationsdokuments finden Sie unter Konfigurationsdokument in der SSM-API-Referenz.

```
New-SSMDocument -Content (Get-Content -Raw "c:\temp\RunShellScript.json") -Name
 "RunShellScript" -DocumentType "Command"
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter CreateDocument AWS Tools for PowerShellCmdlet-Referenz.

`New-SSMMaintenanceWindow`

Das folgende Codebeispiel zeigt die Verwendung. `New-SSMMaintenanceWindow`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein neues Wartungsfenster mit dem angegebenen Namen erstellt, das an jedem Dienstag um 16 Uhr für 4 Stunden läuft, mit einer Frist von 1 Stunde, und das Ziele ohne Zuordnung zulässt.

```
New-SSMMaintenanceWindow -Name "MyMaintenanceWindow" -Duration 4 -Cutoff 1 -
AllowUnassociatedTarget $true -Schedule "cron(0 16 ? * TUE *)"
```

Ausgabe:

```
mw-03eb53e1ea7383998
```

- Einzelheiten zur API finden Sie unter CreateMaintenanceWindow AWS Tools for PowerShellCmdlet-Referenz.

`New-SSMPatchBaseline`

Das folgende Codebeispiel zeigt die Verwendung. `New-SSMPatchBaseline`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Patch-Baseline erstellt, die Patches sieben Tage nach ihrer Veröffentlichung durch Microsoft für verwaltete Instanzen genehmigt, auf denen Windows Server 2019 in einer Produktionsumgebung ausgeführt wird.

```
$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
```

Ausgabe:

```
pb-0z4z6221c4296b23z
```

- Einzelheiten zur API finden Sie unter CreatePatchBaseline AWS Tools for PowerShellCmdlet-Referenz.

`Register-SSMDefaultPatchBaseline`

Das folgende Codebeispiel zeigt die Verwendung. `Register-SSMDefaultPatchBaseline`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Patch-Baseline als Standard-Patch-Baseline registriert.

```
Register-SSMDefaultPatchBaseline -BaselineId "pb-03da896ca3b68b639"
```

Ausgabe:

```
pb-03da896ca3b68b639
```

- Einzelheiten zur API finden Sie unter RegisterDefaultPatchBaseline AWS Tools for PowerShellCmdlet-Referenz.

`Register-SSMPatchBaselineForPatchGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Register-SSMPatchBaselineForPatchGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Patch-Baseline für eine Patch-Gruppe registriert.

```
Register-SSMPatchBaselineForPatchGroup -BaselineId "pb-03da896ca3b68b639" -
PatchGroup "Production"
```

Ausgabe:

```
BaselineId           PatchGroup
----------           ----------
pb-03da896ca3b68b639 Production
```

- Einzelheiten zur API finden Sie unter RegisterPatchBaselineForPatchGroup AWS Tools for PowerShellCmdlet-Referenz.

`Register-SSMTargetWithMaintenanceWindow`

Das folgende Codebeispiel zeigt die Verwendung. `Register-SSMTargetWithMaintenanceWindow`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Instanz mit einem Wartungsfenster registriert.

```
$option1 = @{Key="InstanceIds";Values=@("i-0000293ffd8c57862")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

Ausgabe:

```
d8e47760-23ed-46a5-9f28-927337725398
```

Beispiel 2: In diesem Beispiel werden mehrere Instanzen mit einem Wartungsfenster registriert.

```
$option1 =
 @{Key="InstanceIds";Values=@("i-0000293ffd8c57862","i-0cb2b964d3e14fd9f")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

Ausgabe:

```
6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

Beispiel 3: In diesem Beispiel wird mithilfe von EC2-Tags eine Instance mit einem Wartungsfenster registriert.

```
$option1 = @{Key="tag:Environment";Values=@("Production")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Production Web Servers" -ResourceType "INSTANCE"
```

Ausgabe:

```
2994977e-aefb-4a71-beac-df620352f184
```

- Einzelheiten zur API finden Sie unter RegisterTargetWithMaintenanceWindow AWS Tools for PowerShellCmdlet-Referenz.

`Register-SSMTaskWithMaintenanceWindow`

Das folgende Codebeispiel zeigt die Verwendung. `Register-SSMTaskWithMaintenanceWindow`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Aufgabe mit einem Wartungsfenster unter Verwendung einer Instanz-ID registriert. Die Ausgabe ist die Task-ID.

```
$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
```

Ausgabe:

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

Beispiel 2: In diesem Beispiel wird eine Aufgabe mit einem Wartungsfenster unter Verwendung einer Ziel-ID registriert. Die Ausgabe ist die Task-ID.

```
$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
```

Ausgabe:

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

Beispiel 3: Dieses Beispiel erstellt ein Parameterobjekt für das Run-Befehlsdokument **AWS-RunPowerShellScript** und erstellt eine Aufgabe mit einem bestimmten Wartungsfenster unter Verwendung der Ziel-ID. Die Rückgabeausgabe ist die Aufgaben-ID.

```
$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
```

Ausgabe:

```
f1e2ef34-5678-12e3-456a-12334c5c6cbe
```

Beispiel 4: In diesem Beispiel wird eine AWS Systems Manager Automation-Aufgabe mithilfe eines Dokuments mit dem Namen registriert**`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"
```

- Einzelheiten zur API finden Sie unter RegisterTaskWithMaintenanceWindow AWS Tools for PowerShellCmdlet-Referenz.

`Remove-SSMActivation`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-SSMActivation`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Aktivierung gelöscht. Es erfolgt keine Ausgabe, wenn der Befehl erfolgreich ist.

```
Remove-SSMActivation -ActivationId "08e51e79-1e36-446c-8e63-9458569c1363"
```

- Einzelheiten zur API finden Sie unter DeleteActivation AWS Tools for PowerShellCmdlet-Referenz.

`Remove-SSMAssociation`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-SSMAssociation`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Verknüpfung zwischen einer Instanz und einem Dokument gelöscht. Es erfolgt keine Ausgabe, wenn der Befehl erfolgreich ist.

```
Remove-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

- Einzelheiten zur API finden Sie unter DeleteAssociation AWS Tools for PowerShellCmdlet-Referenz.

`Remove-SSMDocument`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-SSMDocument`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Dokument gelöscht. Es erfolgt keine Ausgabe, wenn der Befehl erfolgreich ist.

```
Remove-SSMDocument -Name "RunShellScript"
```

- Einzelheiten zur API finden Sie unter DeleteDocument AWS Tools for PowerShellCmdlet-Referenz.

`Remove-SSMMaintenanceWindow`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-SSMMaintenanceWindow`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Wartungsfenster entfernt.

```
Remove-SSMMaintenanceWindow -WindowId "mw-06d59c1a07c022145"
```

Ausgabe:

```
mw-06d59c1a07c022145
```

- Einzelheiten zur API finden Sie unter DeleteMaintenanceWindow AWS Tools for PowerShellCmdlet-Referenz.

`Remove-SSMParameter`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-SSMParameter`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Parameter gelöscht. Es erfolgt keine Ausgabe, wenn der Befehl erfolgreich ist.

```
Remove-SSMParameter -Name "helloWorld"
```

- Einzelheiten zur API finden Sie unter DeleteParameter AWS Tools for PowerShellCmdlet-Referenz.

`Remove-SSMPatchBaseline`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-SSMPatchBaseline`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Patch-Baseline gelöscht.

```
Remove-SSMPatchBaseline -BaselineId "pb-045f10b4f382baeda"
```

Ausgabe:

```
pb-045f10b4f382baeda
```

- Einzelheiten zur API finden Sie unter DeletePatchBaseline AWS Tools for PowerShellCmdlet-Referenz.

`Remove-SSMResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-SSMResourceTag`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Tag aus einem Wartungsfenster entfernt. Es erfolgt keine Ausgabe, wenn der Befehl erfolgreich ist.

```
Remove-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -TagKey "Production"
```

- Einzelheiten zur API finden Sie unter RemoveTagsFromResource AWS Tools for PowerShellCmdlet-Referenz.

`Send-SSMCommand`

Das folgende Codebeispiel zeigt die Verwendung. `Send-SSMCommand`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Echo-Befehl auf einer Zielinstanz ausgeführt.

```
Send-SSMCommand -DocumentName "AWS-RunPowerShellScript" -Parameter @{commands =
 "echo helloWorld"} -Target @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
```

Ausgabe:

```
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}
```

Beispiel 2: Dieses Beispiel zeigt, wie ein Befehl ausgeführt wird, der verschachtelte Parameter akzeptiert.

```
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
```

- Einzelheiten zur API finden Sie unter SendCommand AWS Tools for PowerShellCmdlet-Referenz.

`Start-SSMAutomationExecution`

Das folgende Codebeispiel zeigt die Verwendung. `Start-SSMAutomationExecution`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Dokument ausgeführt, das eine Automatisierungsrolle, eine AMI-Quell-ID und eine Amazon EC2 EC2-Instance-Rolle spezifiziert.

```
Start-SSMAutomationExecution -DocumentName AWS-UpdateLinuxAmi -
Parameter @{'AutomationAssumeRole'='arn:aws:iam::123456789012:role/
SSMAutomationRole';'SourceAmiId'='ami-f173cc91';'InstanceIamRole'='EC2InstanceRole'}
```

Ausgabe:

```
3a532a4f-0382-11e7-9df7-6f11185f6dd1
```

- Einzelheiten zur API finden Sie unter StartAutomationExecution AWS Tools for PowerShellCmdlet-Referenz.

`Stop-SSMAutomationExecution`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-SSMAutomationExecution`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Automatisierungsausführung gestoppt. Es erfolgt keine Ausgabe, wenn der Befehl erfolgreich ist.

```
Stop-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"
```

- Einzelheiten zur API finden Sie unter StopAutomationExecution AWS Tools for PowerShellCmdlet-Referenz.

`Stop-SSMCommand`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-SSMCommand`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird versucht, einen Befehl abzubrechen. Wenn der Vorgang erfolgreich ist, erfolgt keine Ausgabe.

```
Stop-SSMCommand -CommandId "9ded293e-e792-4440-8e3e-7b8ec5feaa38"
```

- Einzelheiten zur API finden Sie unter CancelCommand AWS Tools for PowerShellCmdlet-Referenz.

`Unregister-SSMManagedInstance`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-SSMManagedInstance`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Registrierung einer verwalteten Instanz aufgehoben. Es erfolgt keine Ausgabe, wenn der Befehl erfolgreich ist.

```
Unregister-SSMManagedInstance -InstanceId "mi-08ab247cdf1046573"
```

- Einzelheiten zur API finden Sie unter DeregisterManagedInstance AWS Tools for PowerShellCmdlet-Referenz.

`Unregister-SSMPatchBaselineForPatchGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-SSMPatchBaselineForPatchGroup`

Beispiel 1: In diesem Beispiel wird die Registrierung einer Patchgruppe von einer Patch-Baseline aufgehoben.

```
Unregister-SSMPatchBaselineForPatchGroup -BaselineId "pb-045f10b4f382baeda" -
PatchGroup "Production"
```

Ausgabe:

```
BaselineId           PatchGroup
----------           ----------
pb-045f10b4f382baeda Production
```

- Einzelheiten zur API finden Sie unter DeregisterPatchBaselineForPatchGroup AWS Tools for PowerShellCmdlet-Referenz.

`Unregister-SSMTargetFromMaintenanceWindow`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-SSMTargetFromMaintenanceWindow`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Ziel aus einem Wartungsfenster entfernt.

```
Unregister-SSMTargetFromMaintenanceWindow -WindowTargetId "6ab5c208-9fc4-4697-84b7-
b02a6cc25f7d" -WindowId "mw-06cf17cbefcb4bf4f"
```

Ausgabe:

```
WindowId             WindowTargetId
--------             --------------
mw-06cf17cbefcb4bf4f 6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

- Einzelheiten zur API finden Sie unter DeregisterTargetFromMaintenanceWindow AWS Tools for PowerShellCmdlet-Referenz.

`Unregister-SSMTaskFromMaintenanceWindow`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-SSMTaskFromMaintenanceWindow`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Aufgabe aus einem Wartungsfenster entfernt.

```
Unregister-SSMTaskFromMaintenanceWindow -WindowTaskId "f34a2c47-ddfd-4c85-
a88d-72366b69af1b" -WindowId "mw-03a342e62c96d31b0"
```

Ausgabe:

```
WindowId             WindowTaskId
--------             ------------
mw-03a342e62c96d31b0 f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

- Einzelheiten zur API finden Sie unter DeregisterTaskFromMaintenanceWindow AWS Tools for PowerShellCmdlet-Referenz.

`Update-SSMAssociation`

Das folgende Codebeispiel zeigt die Verwendung. `Update-SSMAssociation`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Verknüpfung mit einer neuen Dokumentversion aktualisiert.

```
Update-SSMAssociation -AssociationId "93285663-92df-44cb-9f26-2292d4ecc439" -
DocumentVersion "1"
```

Ausgabe:

```
Name                 : AWS-UpdateSSMAgent
InstanceId           :
Date                 : 3/1/2017 6:22:21 PM
Status.Name          :
```

```
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

- Einzelheiten zur API finden Sie unter UpdateAssociation AWS Tools for PowerShellCmdlet-Referenz.

`Update-SSMAssociationStatus`

Das folgende Codebeispiel zeigt die Verwendung. `Update-SSMAssociationStatus`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der Zuordnungsstatus der Zuordnung zwischen einer Instanz und einem Konfigurationsdokument aktualisiert.

```
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"
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter UpdateAssociationStatus AWS Tools for PowerShellCmdlet-Referenz.

`Update-SSMDocument`

Das folgende Codebeispiel zeigt die Verwendung. `Update-SSMDocument`

Tools für PowerShell

Beispiel 1: Dadurch wird eine neue Version eines Dokuments mit dem aktualisierten Inhalt der von Ihnen angegebenen JSON-Datei erstellt. Das Dokument muss im JSON-Format sein. Sie können die Dokumentversion mit dem Cmdlet „Get-SSMDocumentVersionList“ abrufen.

```
Update-SSMDocument -Name RunShellScript -DocumentVersion "1" -Content (Get-Content -
Raw "c:\temp\RunShellScript.json")
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter UpdateDocumentCmdlet-Referenz.AWS Tools for PowerShell

`Update-SSMDocumentDefaultVersion`

Das folgende Codebeispiel zeigt die Verwendung. `Update-SSMDocumentDefaultVersion`

Tools für PowerShell

Beispiel 1: Dadurch wird die Standardversion eines Dokuments aktualisiert. Sie können die verfügbaren Dokumentversionen mit dem Cmdlet „Get-SSMDocumentVersionList“ abrufen.

```
Update-SSMDocumentDefaultVersion -Name "RunShellScript" -DocumentVersion "2"
```

Ausgabe:

```
DefaultVersion Name
-------------- ----
2              RunShellScript
```

- Einzelheiten zur API finden Sie unter UpdateDocumentDefaultVersionCmdlet-Referenz.AWS Tools for PowerShell

`Update-SSMMaintenanceWindow`

Das folgende Codebeispiel zeigt die Verwendung. `Update-SSMMaintenanceWindow`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der Name eines Wartungsfensters aktualisiert.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Name "My-Renamed-MW"
```

Ausgabe:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

Beispiel 2: Dieses Beispiel aktiviert ein Wartungsfenster.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $true
```

Ausgabe:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
```

```
Name                        : My-Renamed-MW
Schedule                    : cron(0 */30 * * * ? *)
WindowId                    : mw-03eb9db42890fb82d
```

Beispiel 3: In diesem Beispiel wird ein Wartungsfenster deaktiviert.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $false
```

Ausgabe:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : False
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

- Einzelheiten zur API finden Sie unter UpdateMaintenanceWindow AWS Tools for PowerShellCmdlet-Referenz.

`Update-SSMManagedInstanceRole`

Das folgende Codebeispiel zeigt die Verwendung. `Update-SSMManagedInstanceRole`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Rolle einer verwalteten Instanz aktualisiert. Es erfolgt keine Ausgabe, wenn der Befehl erfolgreich ist.

```
Update-SSMManagedInstanceRole -InstanceId "mi-08ab247cdf1046573" -IamRole
 "AutomationRole"
```

- Einzelheiten zur API finden Sie unter UpdateManagedInstanceRole AWS Tools for PowerShellCmdlet-Referenz.

`Update-SSMPatchBaseline`

Das folgende Codebeispiel zeigt die Verwendung. `Update-SSMPatchBaseline`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden einer vorhandenen Patch-Baseline zwei Patches als abgelehnt und ein Patch als genehmigt hinzugefügt.

```
Update-SSMPatchBaseline -BaselineId "pb-03da896ca3b68b639" -RejectedPatch
 "KB2032276","MS10-048" -ApprovedPatch "KB2124261"
```

Ausgabe:

```
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}
```

- Einzelheiten zur API finden Sie unter UpdatePatchBaseline AWS Tools for PowerShellCmdlet-Referenz.

`Write-SSMComplianceItem`

Das folgende Codebeispiel zeigt die Verwendung. `Write-SSMComplianceItem`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein benutzerdefiniertes Compliance-Element für die angegebene verwaltete Instanz geschrieben

```
$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"
```

- Einzelheiten zur API finden Sie unter PutComplianceItems AWS Tools for PowerShellCmdlet-Referenz.

`Write-SSMInventory`

Das folgende Codebeispiel zeigt die Verwendung. `Write-SSMInventory`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden einer Instanz Informationen zum Rack-Standort zugewiesen. Es erfolgt keine Ausgabe, wenn der Befehl erfolgreich ist.

```
$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
```

- Einzelheiten zur API finden Sie unter PutInventory AWS Tools for PowerShellCmdlet-Referenz.

`Write-SSMParameter`

Das folgende Codebeispiel zeigt die Verwendung. `Write-SSMParameter`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Parameter erstellt. Es erfolgt keine Ausgabe, wenn der Befehl erfolgreich ist.

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "helloWorld"
```

Beispiel 2: In diesem Beispiel wird ein Parameter geändert. Es erfolgt keine Ausgabe, wenn der Befehl erfolgreich ist.

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "Good day, Sunshine!" -
Overwrite $true
```

- Einzelheiten zur API finden Sie unter PutParameter AWS Tools for PowerShellCmdlet-Referenz.

Amazon Translate Translate-Beispiele mit Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Translate Aktionen ausführen und allgemeine Szenarien implementieren. AWS Tools for PowerShell

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`ConvertTo-TRNTargetLanguage`

Das folgende Codebeispiel zeigt die Verwendung`ConvertTo-TRNTargetLanguage`.

Tools für PowerShell

Beispiel 1: Konvertiert den angegebenen englischen Text in Französisch. Der zu konvertierende Text kann auch als -Text-Parameter übergeben werden.

```
"Hello World" | ConvertTo-TRNTargetLanguage -SourceLanguageCode en -
TargetLanguageCode fr
```

- Einzelheiten zur API finden Sie unter TranslateText AWS Tools for PowerShellCmdlet-Referenz.

AWS WAFV2 Beispiele für die Verwendung von Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS Tools for PowerShell with Aktionen ausführen und allgemeine Szenarien implementieren AWS WAFV2.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

Aktionen

`New-WAF2WebACL`

Das folgende Codebeispiel zeigt die Verwendung`New-WAF2WebACL`.

Tools für PowerShell

> Beispiel 1: Dieser Befehl erstellt eine neue Web-ACL mit dem Namen „waf-test“. Bitte beachten Sie, dass 'DefaultAction' gemäß der Dokumentation zur Service-API eine erforderliche Eigenschaft ist. Daher sollte der Wert entweder für '- DefaultAction _Allow 'und/oder '- DefaultAction _Block' angegeben werden. Da '- DefaultAction _Allow' und '- DefaultAction _Block 'nicht die erforderlichen Eigenschaften sind, könnte der Wert '@ {}' als Platzhalter verwendet werden, wie im obigen Beispiel gezeigt.

```
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 @{}
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz. CreateWebAclAWS Tools for PowerShell

WorkSpaces Beispiele für die Verwendung von Tools für PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS Tools for PowerShell with Aktionen ausführen und allgemeine Szenarien implementieren WorkSpaces.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kontext finden.

Themen

- Aktionen

Aktionen

`Approve-WKSIpRule`

Das folgende Codebeispiel zeigt die Verwendung`Approve-WKSIpRule`.

Tools für PowerShell

Beispiel 1: Dieses Beispiel fügt Regeln zu einer vorhandenen IP-Gruppe hinzu

```
$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
```

- Einzelheiten zur API finden Sie unter AuthorizeIpRules AWS Tools for PowerShellCmdlet-Referenz.

`Copy-WKSWorkspaceImage`

Das folgende Codebeispiel zeigt die Verwendung. `Copy-WKSWorkspaceImage`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Workspace-Bild mit der angegebenen ID von us-west-2 in die aktuelle Region mit dem Namen "" kopiert CopiedImageTest

```
Copy-WKSWorkspaceImage -Name CopiedImageTest -SourceRegion us-west-2 -SourceImageId
 wsi-djfoedhw6
```

Ausgabe:

```
wsi-456abaqfe
```

- Einzelheiten zur API finden Sie unter CopyWorkspaceImage AWS Tools for PowerShellCmdlet-Referenz.

`Edit-WKSClientProperty`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-WKSClientProperty`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird Reconnection für den Workspaces-Client aktiviert

```
Edit-WKSClientProperty -Region us-west-2 -ClientProperties_ReconnectEnabled
 "ENABLED" -ResourceId d-123414a369
```

- Einzelheiten zur API finden Sie unter ModifyClientProperties AWS Tools for PowerShellCmdlet-Referenz.

`Edit-WKSSelfServicePermission`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-WKSSelfServicePermission`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Self-Service-Berechtigungen aktiviert, um den Rechnertyp zu ändern und die Volume-Größe für das angegebene Verzeichnis zu erhöhen

```
Edit-WKSSelfservicePermission -Region us-west-2 -ResourceId
 d-123454a369 -SelfservicePermissions_ChangeComputeType ENABLED -
SelfservicePermissions_IncreaseVolumeSize ENABLED
```

- Einzelheiten zur API finden Sie unter ModifySelfservicePermissions AWS Tools for PowerShellCmdlet-Referenz.

`Edit-WKSWorkspaceAccessProperty`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-WKSWorkspaceAccessProperty`

Tools für PowerShell

Beispiel 1: Dieses Beispiel aktiviert den Workspace-Zugriff auf Android und Chrome OS für das angegebene Verzeichnis

```
Edit-WKSWorkspaceAccessProperty -Region us-west-2 -ResourceId
 d-123454a369 -WorkspaceAccessProperties_DeviceTypeAndroid ALLOW -
WorkspaceAccessProperties_DeviceTypeChromeOs ALLOW
```

- Einzelheiten zur API finden Sie unter ModifyWorkspaceAccessProperties AWS Tools for PowerShellCmdlet-Referenz.

`Edit-WKSWorkspaceCreationProperty`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-WKSWorkspaceCreationProperty`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden beim Erstellen eines Workspace die Standardwerte „Internetzugriff“ und „Wartungsmodus“ auf „true“ gesetzt

```
Edit-WKSWorkspaceCreationProperty -Region us-west-2 -ResourceId
 d-123454a369 -WorkspaceCreationProperties_EnableInternetAccess $true -
WorkspaceCreationProperties_EnableMaintenanceMode $true
```

- Einzelheiten zur API finden Sie unter ModifyWorkspaceCreationProperties AWS Tools for PowerShellCmdlet-Referenz.

`Edit-WKSWorkspaceProperty`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-WKSWorkspaceProperty`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Eigenschaft Workspace Running Mode für den angegebenen Workspace auf Auto Stop geändert

```
Edit-WKSWorkspaceProperty -WorkspaceId ws-w361s100v -Region us-west-2 -
WorkspaceProperties_RunningMode AUTO_STOP
```

- Einzelheiten zur API finden Sie unter ModifyWorkspaceProperties AWS Tools for PowerShellCmdlet-Referenz.

`Edit-WKSWorkspaceState`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-WKSWorkspaceState`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der Status des angegebenen Workspace auf Verfügbar geändert

```
Edit-WKSWorkspaceState -WorkspaceId ws-w361s100v -Region us-west-2 -WorkspaceState
 AVAILABLE
```

- Einzelheiten zur API finden Sie unter ModifyWorkspaceState AWS Tools for PowerShellCmdlet-Referenz.

`Get-WKSClientProperty`

Das folgende Codebeispiel zeigt die Verwendung. `Get-WKSClientProperty`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Client-Eigenschaften des Workspace Client für das angegebene Verzeichnis abgerufen

```
Get-WKSClientProperty -ResourceId d-223562a123
```

- Einzelheiten zur API finden Sie unter DescribeClientProperties AWS Tools for PowerShellCmdlet-Referenz.

`Get-WKSIpGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Get-WKSIpGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Details der angegebenen IP-Gruppe in der angegebenen Region abgerufen

```
Get-WKSIpGroup -Region us-east-1 -GroupId wsipg-8m1234v45
```

Ausgabe:

```
GroupDesc GroupId         GroupName UserRules
--------- -------         --------- ---------
          wsipg-8m1234v45 TestGroup {Amazon.WorkSpaces.Model.IpRuleItem,
 Amazon.WorkSpaces.Model.IpRuleItem}
```

- Einzelheiten zur API finden Sie unter DescribeIpGroups AWS Tools for PowerShellCmdlet-Referenz.

`Get-WKSTag`

Das folgende Codebeispiel zeigt die Verwendung. `Get-WKSTag`

Tools für PowerShell

Beispiel 1: Dieses Beispiel ruft das Tag für den angegebenen Workspace ab

```
Get-WKSTag -WorkspaceId ws-w361s234r -Region us-west-2
```

Ausgabe:

```
Key         Value
---         -----
auto-delete no
purpose     Workbench
```

- Einzelheiten zur API finden Sie unter DescribeTags AWS Tools for PowerShellCmdlet-Referenz.

`Get-WKSWorkspace`

Das folgende Codebeispiel zeigt die Verwendung. `Get-WKSWorkspace`

Tools für PowerShell

Beispiel 1: Ruft Details WorkSpaces zu all Ihren Kontakten in der Pipeline ab.

```
Get-WKSWorkspace
```

Ausgabe:

```
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
```

Beispiel 2: Dieser Befehl zeigt die Werte der untergeordneten Eigenschaften eines **`WorkSpaceProperties`** Workspace in der **`us-west-2`** Region an. Weitere Informationen zu den untergeordneten Eigenschaften von **`WorkSpaceProperties`** finden Sie unter https://docs.aws.amazon.com/workspaces/latest/api/API_ WorkspaceProperties .html.

```
(Get-WKSWorkspace -Region us-west-2 -WorkSpaceId ws-xdaf7hc9s).WorkspaceProperties
```

Ausgabe:

```
        ComputeTypeName                     : STANDARD
        RootVolumeSizeGib                   : 80
        RunningMode                         : AUTO_STOP
        RunningModeAutoStopTimeoutInMinutes : 60
        UserVolumeSizeGib                   : 50
```

Beispiel 3: Dieser Befehl zeigt den Wert der untergeordneten Eigenschaft **`RootVolumeSizeGib`** von **`WorkSpaceProperties`** für einen Workspace in der **`us-west-2`** Region an. Die Größe des Root-Volumes in GiB beträgt 80.

```
(Get-WKSWorkspace -Region us-west-2 -WorkSpaceId ws-
xdaf7hc9s).WorkspaceProperties.RootVolumeSizeGib
```

Ausgabe:

```
        80
```

- Einzelheiten zur API finden Sie unter DescribeWorkspaces AWS Tools for PowerShellCmdlet-Referenz.

`Get-WKSWorkspaceBundle`

Das folgende Codebeispiel zeigt die Verwendung. `Get-WKSWorkspaceBundle`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Details zu allen Workspace-Bundles in der aktuellen Region abgerufen

```
Get-WKSWorkspaceBundle
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeWorkspaceBundles AWS Tools for PowerShellCmdlet-Referenz.

`Get-WKSWorkspaceDirectory`

Das folgende Codebeispiel zeigt die Verwendung. `Get-WKSWorkspaceDirectory`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Verzeichnisdetails für registrierte Verzeichnisse aufgeführt

```
Get-WKSWorkspaceDirectory
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DescribeWorkspaceDirectories AWS Tools for PowerShellCmdlet-Referenz.

`Get-WKSWorkspaceImage`

Das folgende Codebeispiel zeigt die Verwendung. `Get-WKSWorkspaceImage`

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden alle Details aller Bilder in der Region abgerufen

```
Get-WKSWorkspaceImage
```

Ausgabe:

```
Description     :This image is copied from another image
ErrorCode       :
ErrorMessage    :
ImageId         : wsi-345ahdjgo
Name            : CopiedImageTest
OperatingSystem : Amazon.WorkSpaces.Model.OperatingSystem
RequiredTenancy : DEFAULT
State           : AVAILABLE
```

- Einzelheiten zur API finden Sie unter DescribeWorkspaceImages AWS Tools for PowerShellCmdlet-Referenz.

`Get-WKSWorkspaceSnapshot`

Das folgende Codebeispiel zeigt die Verwendung. `Get-WKSWorkspaceSnapshot`

Tools für PowerShell

Beispiel 1: Dieses Beispiel zeigt den Zeitstempel des zuletzt für den angegebenen Workspace erstellten Snapshots

```
Get-WKSWorkspaceSnapshot -WorkspaceId ws-w361s100v
```

Ausgabe:

```
RebuildSnapshots                    RestoreSnapshots
----------------                    ----------------
{Amazon.WorkSpaces.Model.Snapshot} {Amazon.WorkSpaces.Model.Snapshot}
```

- Einzelheiten zur API finden Sie unter DescribeWorkspaceSnapshots AWS Tools for PowerShellCmdlet-Referenz.

`Get-WKSWorkspacesConnectionStatus`

Das folgende Codebeispiel zeigt die Verwendung. `Get-WKSWorkspacesConnectionStatus`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der Verbindungsstatus für den angegebenen Workspace abgerufen

```
Get-WKSWorkspacesConnectionStatus -WorkspaceId ws-w123s234r
```

- Einzelheiten zur API finden Sie unter DescribeWorkspacesConnectionStatus AWS Tools for PowerShellCmdlet-Referenz.

`New-WKSIpGroup`

Das folgende Codebeispiel zeigt die Verwendung. `New-WKSIpGroup`

Tools für PowerShell

Beispiel 1: Dieses Beispiel erstellt eine leere IP-Gruppe mit dem Namen FreshEmptyIpGroup

```
New-WKSIpGroup -GroupName "FreshNewIPGroup"
```

Ausgabe:

```
wsipg-w45rty4ty
```

- Einzelheiten zur API finden Sie unter CreateIpGroup AWS Tools for PowerShellCmdlet-Referenz.

`New-WKSTag`

Das folgende Codebeispiel zeigt die Verwendung. `New-WKSTag`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein neues Tag zu einem Workspace mit dem Namen hinzugefügt**`ws-wsname`**. Das Tag hat den Schlüssel „Name“ und den Schlüsselwert**`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
```

Beispiel 2: In diesem Beispiel werden mehrere Tags zu einem Arbeitsbereich mit dem Namen hinzugefügt**`ws-wsname`**. Ein Tag hat den Schlüssel „Name“ und den Schlüsselwert**`AWS_Workspace`**; das andere Tag hat den Tag-Schlüssel „Stage“ und den Schlüsselwert „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
```

- Einzelheiten zur API finden Sie unter CreateTags AWS Tools for PowerShellCmdlet-Referenz.

`New-WKSWorkspace`

Das folgende Codebeispiel zeigt die Verwendung. `New-WKSWorkspace`

Tools für PowerShell

Beispiel 1: Erstellen Sie eine WorkSpace für das bereitgestellte Paket, das Verzeichnis und den Benutzer.

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId" =
 "d-1a2b3c4d"; "UserName" = "USERNAME"}
```

Beispiel 2: In diesem Beispiel werden mehrere erstellt WorkSpaces

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId"
 = "d-1a2b3c4d"; "UserName" = "USERNAME_1"},@{"BundleID" = "wsb-1a2b3c4d";
 "DirectoryId" = "d-1a2b3c4d"; "UserName" = "USERNAME_2"}
```

- Einzelheiten zur API finden Sie unter CreateWorkspaces AWS Tools for PowerShellCmdlet-Referenz.

`Register-WKSIpGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Register-WKSIpGroup`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene IP-Gruppe im angegebenen Verzeichnis registriert

```
Register-WKSIpGroup -GroupId wsipg-23ahsdres -DirectoryId d-123412e123
```

- Einzelheiten zur API finden Sie unter AssociateIpGroups AWS Tools for PowerShellCmdlet-Referenz.

`Register-WKSWorkspaceDirectory`

Das folgende Codebeispiel zeigt die Verwendung. `Register-WKSWorkspaceDirectory`

Tools für PowerShell

Beispiel 1: Dieses Beispiel registriert das angegebene Verzeichnis für Workspaces Service

```
Register-WKSWorkspaceDirectory -DirectoryId d-123412a123 -EnableWorkDoc $false
```

- Einzelheiten zur API finden Sie unter RegisterWorkspaceDirectory AWS Tools for PowerShellCmdlet-Referenz.

`Remove-WKSIpGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-WKSIpGroup`

Tools für PowerShell

Beispiel 1: Dieses Beispiel löscht die angegebene IP-Gruppe

```
Remove-WKSIpGroup -GroupId wsipg-32fhgtred
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DeleteIpGroup AWS Tools for PowerShellCmdlet-Referenz.

`Remove-WKSTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-WKSTag`

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das dem Workspace zugeordnete Tag entfernt

```
Remove-WKSTag -ResourceId ws-w10b3abcd -TagKey "Type"
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter DeleteTags AWS Tools for PowerShellCmdlet-Referenz.

`Remove-WKSWorkspace`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-WKSWorkspace`

Tools für PowerShell

Beispiel 1: Beendet mehrere WorkSpaces. Die Verwendung der Option -Force verhindert, dass das Cmdlet zur Bestätigung auffordert.

```
Remove-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0" -Force
```

Beispiel 2: Ruft die Auflistung aller Ihrer IDs ab WorkSpaces und leitet die IDs über die Pipeline an den WorkSpaceId Parameter - von Remove-WksWorkspace, wodurch alle von beendet werden. WorkSpaces Das Cmdlet fragt nach, bevor jedes Cmdlet beendet wird. WorkSpace Um die Bestätigungsaufforderung zu unterdrücken, fügen Sie die Option -Force hinzu.

```
Get-WKSWorkspaces | Remove-WKSWorkspace
```

Beispiel 3: Dieses Beispiel zeigt, wie TerminateRequest Objekte übergeben werden, die definieren, dass sie beendet werden WorkSpaces sollen. Das Cmdlet fordert Sie zur Bestätigung auf, bevor Sie fortfahren, es sei denn, der Switch-Parameter -Force ist ebenfalls angegeben.

```
$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
```

- Einzelheiten zur API finden Sie unter TerminateWorkspacesCmdlet-Referenz.AWS Tools for PowerShell

`Reset-WKSWorkspace`

Das folgende Codebeispiel zeigt die Verwendung. `Reset-WKSWorkspace`

Tools für PowerShell

Beispiel 1: Baut die angegebene WorkSpace Datei neu auf.

```
Reset-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

Beispiel 2: Ruft die Sammlung all Ihrer Daten ab WorkSpaces und leitet die IDs über die Pipeline an den WorkSpaceId Parameter - von Reset-WksWorkspace weiter, wodurch der neu erstellt wird. WorkSpaces

```
Get-WKSWorkspaces | Reset-WKSWorkspace
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz. RebuildWorkspacesAWS Tools for PowerShell

`Restart-WKSWorkspace`

Das folgende Codebeispiel zeigt die Verwendung. `Restart-WKSWorkspace`

Tools für PowerShell

Beispiel 1: Startet den angegebenen WorkSpace neu.

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

Beispiel 2: Startet mehrere neu. WorkSpaces

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d","ws-5a6b7c8d"
```

Beispiel 3: Ruft die Sammlung all Ihrer Daten ab WorkSpaces und leitet die IDs an den WorkSpaceId Parameter - von Restart-WksWorkspace weiter, wodurch der neu gestartet wird. WorkSpaces

```
Get-WKSWorkspaces | Restart-WKSWorkspace
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz. RebootWorkspacesAWS Tools for PowerShell

`Stop-WKSWorkspace`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-WKSWorkspace`

Tools für PowerShell

Beispiel 1: Stoppt mehrere WorkSpaces.

```
Stop-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0"
```

Beispiel 2: Ruft die Sammlung all Ihrer Daten ab WorkSpaces und leitet die IDs an den WorkSpaceId Parameter - von Stop-WksWorkspace weiter, wodurch der gestoppt wird. WorkSpaces

```
Get-WKSWorkspaces | Stop-WKSWorkspace
```

Beispiel 3: Dieses Beispiel zeigt, wie StopRequest Objekte übergeben werden, die definieren, dass sie gestoppt werden sollen. WorkSpaces

```
$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
```

- Einzelheiten zur API finden Sie unter StopWorkspaces AWS Tools for PowerShellCmdlet-Referenz.

`Unregister-WKSIpGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-WKSIpGroup`

Beispiel 1: In diesem Beispiel wird die Registrierung der angegebenen IP-Gruppe im angegebenen Verzeichnis aufgehoben

```
Unregister-WKSIpGroup -GroupId wsipg-12abcdphq -DirectoryId d-123454b123
```

- Einzelheiten zur API finden Sie unter DisassociateIpGroups AWS Tools for PowerShellCmdlet-Referenz.

Sicherheit für dieses AWS Produkt oder diese Dienstleistung

Cloud-Sicherheit genießt bei Amazon Web Services (AWS) höchste Priorität. Als AWS -Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die zur Erfüllung der Anforderungen von Organisationen entwickelt wurden, für die Sicherheit eine kritische Bedeutung hat. Sicherheit ist eine gemeinsame Verantwortung zwischen Ihnen AWS und Ihnen. Im Modell der übergreifenden Verantwortlichkeit wird Folgendes mit „Sicherheit der Cloud“ bzw. „Sicherheit in der Cloud“ umschrieben:

Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, auf der alle in der AWS Cloud angebotenen Dienste ausgeführt werden, und für die Bereitstellung von Diensten, die Sie sicher nutzen können. Unsere Sicherheitsverantwortung hat bei uns höchste Priorität AWS, und die Wirksamkeit unserer Sicherheit wird im Rahmen der AWS Compliance-Programme regelmäßig von externen Prüfern getestet und verifiziert.

Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem von Ihnen genutzten AWS Dienst und anderen Faktoren, wie der Sensibilität Ihrer Daten, den Anforderungen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften.

Dieses AWS Produkt oder dieser Service folgt dem Modell der gemeinsamen Verantwortung in Bezug auf die spezifischen Amazon Web Services (AWS) -Services, die es unterstützt. Informationen zur AWS Servicesicherheit finden Sie auf der Seite mit der Dokumentation zur AWS Servicesicherheit und den AWS Services, für die das AWS Compliance-Programm zur Einhaltung der Vorschriften zuständig ist.

Themen

Datenschutz in diesem Produkt oder Service von AWS

Das Modell der geteilten Verantwortung von AWS gilt für den Datenschutz in diesem Produkt oder Service von AWS. Wie in diesem Modell beschrieben, ist AWS für den Schutz der globalen

Infrastruktur verantwortlich, in der die gesamte AWS Cloud ausgeführt wird. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie unter Häufig gestellte Fragen zum Datenschutz. Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag AWS-Modell der geteilten Verantwortung und in der DSGVO im AWS-Sicherheitsblog.

Aus Datenschutzgründen empfehlen wir, AWS-Konto-Anmeldeinformationen zu schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einzurichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor Authentifizierung (MFA).
- Verwenden Sie SSL/TLS für die Kommunikation mit AWS-Ressourcen. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit AWS CloudTrail ein.
- Verwenden Sie AWS-Verschlüsselungslösungen zusammen mit allen Standardsicherheitskontrollen in AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff auf AWS über eine Befehlszeilenschnittstelle oder über eine API FIPS 140-2-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-2.

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit diesem Produkt oder Service von AWS oder anderen AWS-Services arbeiten, die die Konsole, die API, die AWS CLI oder AWS SDKs verwenden. Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL für einen externen Server bereitstellen, empfehlen wir dringend, keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

Datenverschlüsselung

Ein wesentliches Merkmal eines sicheren Service ist, dass Informationen verschlüsselt werden, wenn sie nicht aktiv verwendet werden.

Verschlüsselung im Ruhezustand

Die AWS Tools for PowerShell speichert selbst keine Kundendaten außer den Anmeldeinformationen, die sie für die Interaktion mit den AWS-Services im Namen des Benutzers benötigt.

Wenn Sie die AWS Tools for PowerShell verwenden, um einen AWS-Service aufzurufen, der Kundendaten zur Speicherung an Ihren lokalen Computer übermittelt, finden Sie im Kapitel „Security & Compliance“ (Sicherheit und Compliance) im Benutzerhandbuch dieses Service Informationen darüber, wie diese Daten gespeichert, geschützt und verschlüsselt werden.

Verschlüsselung während der Übertragung

Standardmäßig werden alle Daten, die von dem Client-Computer mit den Service-Endpunkten AWS Tools for PowerShell und AWS übertragen werden, verschlüsselt, indem alles über eine HTTPS/TLS-Verbindung gesendet wird.

Sie müssen nichts tun, um die Verwendung von HTTPS/TLS zu aktivieren. Es ist immer aktiviert.

Identitäts- und Zugriffsverwaltung

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service , den Zugriff auf Ressourcen sicher zu AWS kontrollieren. IAM-Administratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert werden kann (über Berechtigungen verfügt), um Ressourcen zu verwenden. AWS IAM ist ein Programm AWS-Service , das Sie ohne zusätzliche Kosten nutzen können.

Themen

- Fehlerbehebung bei AWS Identität und Zugriff

Zielgruppe

Die Art und Weise, wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von der Arbeit ab, in der Sie tätig sind. AWS

Dienstbenutzer — Wenn Sie dies AWS-Services für Ihre Arbeit verwenden, stellt Ihnen Ihr Administrator die erforderlichen Anmeldeinformationen und Berechtigungen zur Verfügung. Wenn Sie für Ihre Arbeit mehr AWS Funktionen verwenden, benötigen Sie möglicherweise zusätzliche Berechtigungen. Wenn Sie die Fuktionsweise der Zugriffskontrolle nachvollziehen, wissen Sie bereits, welche Berechtigungen Sie von Ihrem Administrator anzufordern müssen. Falls Sie auf eine Funktion nicht zugreifen können AWS, finden Fehlerbehebung bei AWS Identität und Zugriff Sie weitere Informationen in der Bedienungsanleitung der von AWS-Service Ihnen verwendeten.

Serviceadministrator — Wenn Sie in Ihrem Unternehmen für die AWS Ressourcen verantwortlich sind, haben Sie wahrscheinlich vollen Zugriff auf AWS. Es ist Ihre Aufgabe, zu bestimmen, auf welche AWS Funktionen und Ressourcen Ihre Servicebenutzer zugreifen sollen. Sie müssen dann Anträge an Ihren IAM-Administrator stellen, um die Berechtigungen Ihrer Servicenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die Grundkonzepte von IAM nachzuvollziehen. Weitere Informationen darüber, wie Ihr Unternehmen IAM verwenden kann AWS, finden Sie in der Benutzeranleitung des von AWS-Service Ihnen verwendeten.

IAM-Administrator: Wenn Sie als IAM-Administrator fungieren, sollten Sie Einzelheiten dazu kennen, wie Sie Richtlinien zur Verwaltung des Zugriffs auf AWS verfassen können. Beispiele für AWS identitätsbasierte Richtlinien, die Sie in IAM verwenden können, finden Sie im Benutzerhandbuch der AWS-Service von Ihnen verwendeten.

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen als IAM-Benutzer authentifiziert (angemeldet AWS) sein oder eine IAM-Rolle annehmen. Root-Benutzer des AWS-Kontos

Sie können sich AWS als föderierte Identität anmelden, indem Sie Anmeldeinformationen verwenden, die über eine Identitätsquelle bereitgestellt wurden. AWS IAM Identity Center (IAM Identity Center) -Benutzer, die Single Sign-On-Authentifizierung Ihres Unternehmens und Ihre Google- oder Facebook-Anmeldeinformationen sind Beispiele für föderierte Identitäten. Wenn Sie sich als Verbundidentität anmelden, hat der Administrator vorher mithilfe von IAM-Rollen einen

Identitätsverbund eingerichtet. Wenn Sie über den Verbund darauf zugreifen AWS , übernehmen Sie indirekt eine Rolle.

Je nachdem, welcher Benutzertyp Sie sind, können Sie sich beim AWS Management Console oder beim AWS Zugangsportal anmelden. Weitere Informationen zur Anmeldung finden Sie AWS unter So melden Sie sich bei Ihrem an AWS-Konto im AWS-Anmeldung Benutzerhandbuch.

Wenn Sie AWS programmgesteuert zugreifen, AWS stellt es ein Software Development Kit (SDK) und eine Befehlszeilenschnittstelle (CLI) bereit, um Ihre Anfragen mithilfe Ihrer Anmeldeinformationen kryptografisch zu signieren. Wenn Sie keine AWS Tools verwenden, müssen Sie Anfragen selbst signieren. Weitere Informationen zur Verwendung der empfohlenen Methode, um Anfragen selbst zu signieren, finden Sie im IAM-Benutzerhandbuch unter AWS API-Anfragen signieren.

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen angeben. AWS Empfiehlt beispielsweise, die Multi-Faktor-Authentifizierung (MFA) zu verwenden, um die Sicherheit Ihres Kontos zu erhöhen. Weitere Informationen finden Sie unter Multi-Faktor-Authentifizierung im AWS IAM Identity Center - Benutzerhandbuch und Verwenden der Multi-Faktor-Authentifizierung (MFA) in AWS im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, die vollständigen Zugriff auf alle AWS-Services Ressourcen im Konto hat. Diese Identität wird als AWS-Konto Root-Benutzer bezeichnet. Sie können darauf zugreifen, indem Sie sich mit der E-Mail-Adresse und dem Passwort anmelden, mit denen Sie das Konto erstellt haben. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Schützen Sie Ihre Root-Benutzer-Anmeldeinformationen und verwenden Sie diese, um die Aufgaben auszuführen, die nur der Root-Benutzer ausführen kann. Eine vollständige Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Aufgaben, die Root-Benutzer-Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Verbundidentität

Als bewährte Methode sollten menschliche Benutzer, einschließlich Benutzer, die Administratorzugriff benötigen, für den Zugriff AWS-Services mithilfe temporärer Anmeldeinformationen den Verbund mit einem Identitätsanbieter verwenden.

Eine föderierte Identität ist ein Benutzer aus Ihrem Unternehmensbenutzerverzeichnis, einem Web-Identitätsanbieter AWS Directory Service, dem Identity Center-Verzeichnis oder einem

beliebigen Benutzer, der mithilfe AWS-Services von Anmeldeinformationen zugreift, die über eine Identitätsquelle bereitgestellt wurden. Wenn föderierte Identitäten darauf zugreifen AWS-Konten, übernehmen sie Rollen, und die Rollen stellen temporäre Anmeldeinformationen bereit.

Für die zentrale Zugriffsverwaltung empfehlen wir Ihnen, AWS IAM Identity Center zu verwenden. Sie können Benutzer und Gruppen in IAM Identity Center erstellen, oder Sie können eine Verbindung zu einer Gruppe von Benutzern und Gruppen in Ihrer eigenen Identitätsquelle herstellen und diese synchronisieren, um sie in all Ihren AWS-Konten Anwendungen zu verwenden. Informationen zu IAM Identity Center finden Sie unter Was ist IAM Identity Center? im AWS IAM Identity Center - Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität innerhalb Ihres Unternehmens AWS-Konto , die über spezifische Berechtigungen für eine einzelne Person oder Anwendung verfügt. Wenn möglich, empfehlen wir, temporäre Anmeldeinformationen zu verwenden, anstatt IAM-Benutzer zu erstellen, die langfristige Anmeldeinformationen wie Passwörter und Zugriffsschlüssel haben. Bei speziellen Anwendungsfällen, die langfristige Anmeldeinformationen mit IAM-Benutzern erfordern, empfehlen wir jedoch, die Zugriffsschlüssel zu rotieren. Weitere Informationen finden Sie unter Regelmäßiges Rotieren von Zugriffsschlüsseln für Anwendungsfälle, die langfristige Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Eine IAM-Gruppe ist eine Identität, die eine Sammlung von IAM-Benutzern angibt. Sie können sich nicht als Gruppe anmelden. Mithilfe von Gruppen können Sie Berechtigungen für mehrere Benutzer gleichzeitig angeben. Gruppen vereinfachen die Verwaltung von Berechtigungen, wenn es zahlreiche Benutzer gibt. Sie könnten beispielsweise einer Gruppe mit dem Namen IAMAdmins Berechtigungen zum Verwalten von IAM-Ressourcen erteilen.

Benutzer unterscheiden sich von Rollen. Ein Benutzer ist einer einzigen Person oder Anwendung eindeutig zugeordnet. Eine Rolle kann von allen Personen angenommen werden, die sie benötigen. Benutzer besitzen dauerhafte Anmeldeinformationen. Rollen stellen temporäre Anmeldeinformationen bereit. Weitere Informationen finden Sie unter Erstellen eines IAM-Benutzers (anstatt einer Rolle) im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität innerhalb Ihres Unternehmens AWS-Konto , die über bestimmte Berechtigungen verfügt. Sie ist einem IAM-Benutzer vergleichbar, ist aber nicht mit einer bestimmten Person verknüpft. Sie können vorübergehend eine IAM-Rolle in der übernehmen, AWS Management

Console indem Sie die Rollen wechseln. Sie können eine Rolle übernehmen, indem Sie eine AWS CLI oder AWS API-Operation aufrufen oder eine benutzerdefinierte URL verwenden. Weitere Informationen zu Methoden für die Verwendung von Rollen finden Sie unter Verwenden von IAM-Rollen im IAM-Benutzerhandbuch.

IAM-Rollen mit temporären Anmeldeinformationen sind in folgenden Situationen hilfreich:

- Verbundbenutzerzugriff – Um einer Verbundidentität Berechtigungen zuzuweisen, erstellen Sie eine Rolle und definieren Berechtigungen für die Rolle. Wird eine Verbundidentität authentifiziert, so wird die Identität der Rolle zugeordnet und erhält die von der Rolle definierten Berechtigungen. Informationen zu Rollen für den Verbund finden Sie unter Erstellen von Rollen für externe Identitätsanbieter im IAM-Benutzerhandbuch. Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Wenn Sie steuern möchten, worauf Ihre Identitäten nach der Authentifizierung zugreifen können, korreliert IAM Identity Center den Berechtigungssatz mit einer Rolle in IAM. Informationen zu Berechtigungssätzen finden Sie unter Berechtigungssätze im AWS IAM Identity Center -Benutzerhandbuch.
- Temporäre IAM-Benutzerberechtigungen – Ein IAM-Benutzer oder eine -Rolle kann eine IAM-Rolle übernehmen, um vorübergehend andere Berechtigungen für eine bestimmte Aufgabe zu erhalten.
- Kontoübergreifender Zugriff – Sie können eine IAM-Rolle verwenden, um einem vertrauenswürdigen Prinzipal in einem anderen Konto den Zugriff auf Ressourcen in Ihrem Konto zu ermöglichen. Rollen stellen die primäre Möglichkeit dar, um kontoübergreifendem Zugriff zu gewähren. Bei einigen können Sie AWS-Services jedoch eine Richtlinie direkt an eine Ressource anhängen (anstatt eine Rolle als Proxy zu verwenden). Informationen zum Unterschied zwischen Rollen und ressourcenbasierten Richtlinien für den kontenübergreifenden Zugriff finden Sie unter Kontenübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.
- Serviceübergreifender Zugriff — Einige verwenden Funktionen in anderen. AWS-Services AWS-Services Wenn Sie beispielsweise einen Aufruf in einem Service tätigen, führt dieser Service häufig Anwendungen in Amazon-EC2 aus oder speichert Objekte in Amazon-S3. Ein Dienst kann dies mit den Berechtigungen des aufrufenden Prinzipals mit einer Servicerolle oder mit einer serviceverknüpften Rolle tun.
  - Forward Access Sessions (FAS) — Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle verwenden, um Aktionen auszuführen AWS, gelten Sie als Principal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service initiieren. FAS verwendet die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, in Kombination mit der Anfrage, Anfragen an AWS-Service nachgelagerte Dienste zu stellen. FAS-Anfragen werden nur gestellt, wenn ein Dienst eine Anfrage erhält, für deren Abschluss Interaktionen

mit anderen AWS-Services oder Ressourcen erforderlich sind. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Zugriffssitzungen weiterleiten.

- Servicerolle – Eine Servicerolle ist eine IAM-Rolle, die ein Service übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.
- Dienstbezogene Rolle — Eine dienstbezogene Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS-Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Servicebezogene Rollen erscheinen in Ihrem Dienst AWS-Konto und gehören dem Dienst. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.

- Anwendungen, die auf Amazon EC2 ausgeführt werden — Sie können eine IAM-Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2-Instance ausgeführt werden und API-Anfragen stellen AWS CLI . AWS Das ist eher zu empfehlen, als Zugriffsschlüssel innerhalb der EC2-Instance zu speichern. Um einer EC2-Instance eine AWS Rolle zuzuweisen und sie allen ihren Anwendungen zur Verfügung zu stellen, erstellen Sie ein Instance-Profil, das an die Instance angehängt ist. Ein Instance-Profil enthält die Rolle und ermöglicht, dass Programme, die in der EC2-Instance ausgeführt werden, temporäre Anmeldeinformationen erhalten. Weitere Informationen finden Sie unter Verwenden einer IAM-Rolle zum Erteilen von Berechtigungen für Anwendungen, die auf Amazon-EC2-Instances ausgeführt werden im IAM-Benutzerhandbuch.

Informationen dazu, wann Sie IAM-Rollen oder IAM-Benutzer verwenden sollten, finden Sie unter Erstellen einer IAM-Rolle (anstatt eines Benutzers) im IAM-Benutzerhandbuch.

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie ist ein Objekt, AWS das, wenn es einer Identität oder Ressource zugeordnet ist, deren Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Prinzipal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anfrage stellt. Berechtigungen in den Richtlinien bestimmen, ob die Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu Struktur und Inhalten von JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

IAM-Richtlinien definieren Berechtigungen für eine Aktion unabhängig von der Methode, die Sie zur Ausführung der Aktion verwenden. Angenommen, es gibt eine Richtlinie, die Berechtigungen für die `iam:GetRole`-Aktion erteilt. Ein Benutzer mit dieser Richtlinie kann Rolleninformationen von der AWS Management Console AWS CLI, der oder der AWS API abrufen.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können weiter als Inline-Richtlinien oder verwaltete Richtlinien kategorisiert werden. Inline-Richtlinien sind direkt in einen einzelnen Benutzer, eine einzelne Gruppe oder eine einzelne Rolle eingebettet. Verwaltete Richtlinien sind eigenständige Richtlinien, die Sie mehreren Benutzern, Gruppen und Rollen in Ihrem System zuordnen können AWS-Konto. Zu den verwalteten Richtlinien gehören AWS verwaltete Richtlinien und vom Kunden verwaltete Richtlinien. Informationen dazu, wie Sie zwischen einer verwalteten Richtlinie und einer eingebundenen Richtlinie wählen, finden Sie unter Auswahl zwischen verwalteten und eingebundenen Richtlinien im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen

in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Zugriffssteuerungslisten (ACLs)

Zugriffssteuerungslisten (ACLs) steuern, welche Prinzipale (Kontomitglieder, Benutzer oder Rollen) auf eine Ressource zugreifen können. ACLs sind ähnlich wie ressourcenbasierte Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Amazon S3 und Amazon VPC sind Beispiele für Services, die ACLs unterstützen. AWS WAF Weitere Informationen“ zu ACLs finden Sie unter Zugriffskontrollliste (ACL) – Übersicht (Access Control List) im Amazon-Simple-Storage-Service-Entwicklerhandbuch.

Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger verbreitete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- Berechtigungsgrenzen – Eine Berechtigungsgrenze ist ein erweitertes Feature, mit der Sie die maximalen Berechtigungen festlegen können, die eine identitätsbasierte Richtlinie einer IAM-Entität (IAM-Benutzer oder -Rolle) erteilen kann. Sie können eine Berechtigungsgrenze für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch Berechtigungsgrenzen eingeschränkt. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen über Berechtigungsgrenzen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im IAM-Benutzerhandbuch.
- Service Control Policies (SCPs) — SCPs sind JSON-Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OU) in festlegen. AWS Organizations AWS Organizations ist ein Dienst zur Gruppierung und zentralen Verwaltung mehrerer Objekte AWS-Konten , die Ihrem Unternehmen gehören. Wenn Sie innerhalb einer Organisation alle Features aktivieren, können Sie Service-Kontrollrichtlinien (SCPs) auf alle oder einzelne Ihrer Konten anwenden. Das SCP schränkt die Berechtigungen für Entitäten in Mitgliedskonten ein, einschließlich der einzelnen Entitäten. Root-Benutzer des AWS-Kontos

Weitere Informationen zu Organizations und SCPs finden Sie unter Funktionsweise von SCPs im AWS Organizations -Benutzerhandbuch.

- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn mehrere auf eine Anforderung mehrere Richtlinientypen angewendet werden können, sind die entsprechenden Berechtigungen komplizierter. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie im IAM-Benutzerhandbuch unter Bewertungslogik für Richtlinien.

Wie AWS-Services arbeiten Sie mit IAM

Einen allgemeinen Überblick darüber, wie die meisten IAM-Funktionen AWS-Services funktionieren, finden Sie im AWS IAM-Benutzerhandbuch unter Dienste, die mit IAM funktionieren.

Informationen zur Verwendung bestimmter Dienste AWS-Service mit IAM finden Sie im Abschnitt Sicherheit im Benutzerhandbuch des jeweiligen Dienstes.

Fehlerbehebung bei AWS Identität und Zugriff

Verwenden Sie die folgenden Informationen, um häufig auftretende Probleme zu diagnostizieren und zu beheben, die bei der Arbeit mit AWS und IAM auftreten können.

Themen

- Ich bin nicht berechtigt, eine Aktion durchzuführen in AWS
- Ich bin nicht berechtigt, iam auszuführen: PassRole
- Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine AWS Ressourcen ermöglichen

Ich bin nicht berechtigt, eine Aktion durchzuführen in AWS

Wenn Sie eine Fehlermeldung erhalten, dass Sie nicht zur Durchführung einer Aktion berechtigt sind, müssen Ihre Richtlinien aktualisiert werden, damit Sie die Aktion durchführen können.

Der folgende Beispielfehler tritt auf, wenn der IAM-Benutzer `mateojackson` versucht, über die Konsole Details zu einer fiktiven *`my-example-widget`*-Ressource anzuzeigen, jedoch nicht über `awes:`*`GetWidget`*-Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes:GetWidget on resource: my-example-widget
```

In diesem Fall muss die Richtlinie für den Benutzer `mateojackson` aktualisiert werden, damit er mit der `awes:`*`GetWidget`*-Aktion auf die *`my-example-widget`*-Ressource zugreifen kann.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich bin nicht berechtigt, iam auszuführen: PassRole

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht zum Durchführen der `iam:PassRole`-Aktion autorisiert sind, müssen Ihre Richtlinien aktualisiert werden, um eine Rolle an AWSübergeben zu können.

Einige AWS-Services ermöglichen es Ihnen, eine bestehende Rolle an diesen Dienst zu übergeben, anstatt eine neue Servicerolle oder eine dienstverknüpfte Rolle zu erstellen. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in AWS auszuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine AWS Ressourcen ermöglichen

Sie können eine Rolle erstellen, die Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation für den Zugriff auf Ihre Ressourcen verwenden können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Im Fall von Diensten, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (Access Control Lists, ACLs) verwenden, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen darüber, ob diese Funktionen AWS unterstützt werden, finden Sie unterWie AWS-Services arbeiten Sie mit IAM.
- Informationen dazu, wie Sie Zugriff auf Ihre Ressourcen gewähren können, AWS-Konten die Ihnen gehören, finden Sie im IAM-Benutzerhandbuch unter Gewähren des Zugriffs auf einen IAM-Benutzer in einem anderen AWS-Konto , den Sie besitzen.
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren können AWS-Konten, finden Sie AWS-Konten im IAM-Benutzerhandbuch unter Gewähren des Zugriffs für Dritte.
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter Gewähren von Zugriff für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie im IAM-Benutzerhandbuch unter Kontenübergreifender Ressourcenzugriff in IAM.

Konformitätsvalidierung für dieses AWS Produkt oder diese Dienstleistung

Informationen darüber, ob AWS-Service ein AWS-Services in den Geltungsbereich bestimmter Compliance-Programme fällt, finden Sie unter Umfang nach Compliance-Programm AWS-Services unter . Wählen Sie dort das Compliance-Programm aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter AWS Compliance-Programme AWS .

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter Berichte herunterladen unter .

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. AWS stellt die folgenden Ressourcen zur Verfügung, die Sie bei der Einhaltung der Vorschriften unterstützen:

- Schnellstartanleitungen zu Sicherheit und Compliance — In diesen Bereitstellungsleitfäden werden architektonische Überlegungen erörtert und Schritte für die Bereitstellung von Basisumgebungen beschrieben AWS , bei denen Sicherheit und Compliance im Mittelpunkt stehen.
- Architecting for HIPAA Security and Compliance on Amazon Web Services — In diesem Whitepaper wird beschrieben, wie Unternehmen HIPAA-fähige Anwendungen erstellen AWS können.

  > **Note**
  >
  > AWS-Services Nicht alle sind HIPAA-fähig. Weitere Informationen finden Sie in der Referenz für HIPAA-berechtigte Services.

- AWS Compliance-Ressourcen — Diese Sammlung von Arbeitsmappen und Leitfäden gilt möglicherweise für Ihre Branche und Ihren Standort.
- AWS Leitfäden zur Einhaltung von Vorschriften für Kunden — Verstehen Sie das Modell der gemeinsamen Verantwortung aus dem Blickwinkel der Einhaltung von Vorschriften. In den Leitfäden werden die bewährten Verfahren zur Sicherung zusammengefasst AWS-Services und die Leitlinien den Sicherheitskontrollen in verschiedenen Frameworks (einschließlich des National Institute of Standards and Technology (NIST), des Payment Card Industry Security Standards Council (PCI) und der International Organization for Standardization (ISO)) zugeordnet.
- Evaluierung von Ressourcen anhand von Regeln im AWS Config Entwicklerhandbuch — Der AWS Config Service bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- AWS Security Hub— Dies AWS-Service bietet einen umfassenden Überblick über Ihren internen Sicherheitsstatus. AWS Security Hub verwendet Sicherheitskontrollen, um Ihre AWS -Ressourcen zu bewerten und Ihre Einhaltung von Sicherheitsstandards und bewährten Methoden zu überprüfen. Eine Liste der unterstützten Services und Kontrollen finden Sie in der Security-Hub-Steuerungsreferenz.
- Amazon GuardDuty — Dies AWS-Service erkennt potenzielle Bedrohungen für Ihre Workloads AWS-Konten, Container und Daten, indem es Ihre Umgebung auf verdächtige und böswillige Aktivitäten überwacht. GuardDuty kann Ihnen helfen, verschiedene Compliance-Anforderungen

wie PCI DSS zu erfüllen, indem es die in bestimmten Compliance-Frameworks vorgeschriebenen Anforderungen zur Erkennung von Eindringlingen erfüllt.

- AWS Audit Manager— Auf diese AWS-Service Weise können Sie Ihre AWS Nutzung kontinuierlich überprüfen, um das Risikomanagement und die Einhaltung von Vorschriften und Industriestandards zu vereinfachen.

Dieses AWS Produkt oder dieser Service folgt dem Modell der gemeinsamen Verantwortung in Bezug auf die spezifischen Amazon Web Services (AWS) -Services, die es unterstützt. Informationen zur AWS Servicesicherheit finden Sie auf der Seite mit der Dokumentation zur AWS Servicesicherheit und den AWS Services, für die das AWS Compliance-Programm zur Einhaltung der Vorschriften zuständig ist.

Erzwingen einer TLS-Mindestversion in den Tools für PowerShell

Für mehr Sicherheit bei der Kommunikation mit AWS-Services sollten Sie die Tools für PowerShell so konfigurieren, dass sie die entsprechende TLS-Version verwenden. Informationen dazu finden Sie unter Erzwingen einer TLS-Mindestversion im Entwicklerhandbuch für AWS SDK for .NET.

Zusätzliche Sicherheitsüberlegungen für die Tools für PowerShell

Dieses Thema enthält zusätzlich zu den Sicherheitsthemen, die in den vorherigen Abschnitten behandelt wurden, Sicherheitsüberlegungen.

Protokollierung vertraulicher Informationen

Bei einigen Vorgängen dieses Tools werden möglicherweise Informationen zurückgegeben, die als vertraulich angesehen werden könnten, einschließlich Informationen aus Umgebungsvariablen. Die Offenlegung dieser Informationen kann in bestimmten Szenarien ein Sicherheitsrisiko darstellen. Beispielsweise könnten die Informationen in CI/CD-Protokollen (Continuous Integration and Continuous Deployment) enthalten sein. Es ist daher wichtig, dass Sie überprüfen, wann Sie solche Ausgaben in Ihre Protokolle aufnehmen, und die Ausgabe unterdrücken, wenn sie nicht benötigt werden. Weitere Informationen zum Schutz sensibler Daten finden Sie unterDatenschutz in diesem Produkt oder Service von AWS.

Beachten Sie die folgenden bewährten Methoden:

- Verwenden Sie keine Umgebungsvariablen, um sensible Werte für Ihre serverlosen Ressourcen zu speichern. Lassen Sie stattdessen Ihren serverlosen Code das Geheimnis programmgesteuert aus einem Geheimspeicher abrufen (z. B.). AWS Secrets Manager
- Überprüfen Sie den Inhalt Ihrer Build-Logs, um sicherzustellen, dass sie keine vertraulichen Informationen enthalten. Erwägen Sie Ansätze wie die Weiterleitung an /dev/null oder das Erfassen der Ausgabe als Bash oder PowerShell Variable, um Befehlsausgaben zu unterdrücken.
- Berücksichtigen Sie den Zugriff auf Ihre Logs und legen Sie den Umfang des Zugriffs entsprechend Ihrem Anwendungsfall fest.

Cmdslet-Referenz für die Tools für PowerShell

Die Tools für PowerShell stellen Cmdlets bereit, mit denen Sie auf AWS-Services zugreifen können. in der AWS Tools for PowerShell-Cmdlet-Referenz finden Sie, welche Cmdlets verfügbar sind.

Dokumentverlauf

In diesem Thema werden signifikante Änderungen der Dokumentation zu AWS Tools for PowerShell beschrieben.

Wir überarbeiten die Dokumentation auch regelmäßig als Reaktion auf Feedback von Kunden. Um Ihr Feedback zu einem Thema einzusenden, verwenden Sie die Feedback-Schaltflächen neben „Hat Ihnen diese Seite geholfen?“, die sich unten auf jeder Seite befinden.

Weitere Informationen zu Änderungen und Aktualisierungen von finden Sie in den AWS Tools for PowerShellVersionshinweisen.

Änderung	Beschreibung	Datum
Codebeispiele	Es wurde ein Kapitel mit Cmdlet-Beispielen hinzugefügt.	17. April 2024
Zusätzliche Sicherheitsüberlegungen	Enthält Informationen zur möglichen Protokollierung sensibler Daten.	16. April 2024
Konfigurieren Sie die Tool-Authentifizierung mit AWS	Es wurden Informationen zur Unterstützung von SSO in der hinzugefügt AWS Tools for PowerShell.	15. März 2024
Cmdlet-Referenz für die Tools für PowerShell	Es wurde ein Abschnitt mit einem Link zur PowerShell Cmdlet-Referenz „Tools for“ hinzugefügt.	17. November 2023
Weitere Updates zu bewährten Methoden für IAM enthalten	Aktualisierter Leitfaden, angepasst an die bewährten IAM-Methoden. Weitere Informationen finden Sie unter Bewährte IAM-Methoden.	12. Oktober 2023

Installation unter Windows	Informationen zur Installation der Tools für Windows PowerShell mithilfe der MSI, die inzwischen veraltet sind, wurden entfernt.	25. September 2023
Aktualisierungen der bewährten Methoden für IAM	Aktualisierter Leitfaden, angepasst an die bewährten IAM-Methoden. Weitere Informationen finden Sie unter Bewährte IAM-Methoden.	8. September 2023
Pipelining und $ AWSHistory	Der `IncludeSensitiveData` -Parameter wurde dem `Set-AWSHistoryConfiguration` -Cmdlet hinzugefügt.	9. März 2023
Verwenden des ClientConfig Parameters in Cmdlets	Es wurden Informationen zur Unterstützung des ClientConfig Parameters hinzugefügt.	28. Oktober 2022
Starten Sie eine Amazon EC2 EC2-Instance mit Windows PowerShell	Hinweise zur Außerbetriebnahme von EC2-Classic hinzugefügt.	26. Juli 2022
AWS Tools for PowerShell Version 4	Es wurden Informationen über Version 4, einschließlich Installationsanweisungen für Windows und Linux/macOS und ein Migrationsthema hinzugefügt, das die Unterschiede zur Version 3 beschreibt und neue Funktionen vorstellt.	21. November 2019

AWS Tools for PowerShell 3.3.563	Es wurden Informationen zur Installation und Verwendung der Vorschauversion des Moduls `AWS.Tools.Common` hinzugefügt. Dieses neue Modul zerlegt das ältere monolithische Paket in ein gemeinsam genutztes Modul und ein Modul pro Dienst. AWS	18. Oktober 2019
AWS Tools for PowerShell 3.3.343.0	Dem AWS Tools for PowerShell Abschnitt „Verwenden“ wurden Informationen zur Einführung der AWS Lambda Tools für PowerShell Core-Entwickler zur PowerShell Erstellung von Funktionen hinzugefügt. AWS Lambda	11. September 2018
AWS Tools for Windows PowerShell 3.1.31.0	In den Abschnitt Erste Schritte wurden Informationen zu neuen Cmdlets eingefügt, die SAML (Security Assertion Markup Language) verwenden, um das Konfigurieren von Verbundidentitäten für Benutzer zu unterstützen.	1. Dezember 2015

AWS Tools for Windows PowerShell 2.3.19	Dem Abschnitt „Cmdlets Discovery and Aliases“ wurden Informationen über das neue `Get-AWSCmdletName` Cmdlet hinzugefügt, mit denen Benutzer ihre gewünscht en Cmdlets leichter finden können. AWS	5. Februar 2015

AWS Tools for Windows PowerShell 1.1.1.0	Die Sammlungsausgabe von Cmdlets wird immer in der Pipeline aufgezähl t. PowerShell Automatis cher Support für auslagerb are Service-Aufrufe Die neue AWSHistory Shell-Var iable $ sammelt Servicean tworten und optional Servicean fragen. AWSRegionInstanzen verwenden das Feld Region, anstatt SystemName das Pipelining zu unterstützen. Remove-S3Bucketunterstützt die Option - DeleteObjects switch. Das Problem mit der Benutzerfreundlichkeit von Set- wurde behobenAW SCredentials. Initialisieren — AWSDefaults meldet, woher es Anmeldeinformationen und Regionsdaten bezogen hat. Stop-EC2Instanceak zeptiert Amazon.EC2.Model.R eservation-Instances als Eingabe. Generic List<T>-P arametertypen durch Array-Typen (T[]) ersetzt. Cmdlets, die Ressourcen löschen oder beenden, fordern vor dem Löschen eine Bestätigu ng an. Write-S3Objectunte rstützt Inline-Textinhalte zum Hochladen auf Amazon S3.	15. Mai 2013

AWS Tools for Windows PowerShell 1.0.1.0	Der Installationsort des Tools für PowerShell Windows-Moduls wurde geändert, sodass Umgebungen, die Windows PowerShell Version 3 verwenden, die Vorteile des automatischen Ladens nutzen können. Das Modul und die zugehörigen Dateien werden jetzt im Unterordner `AWSPowerShell` unter `AWS ToolsPowerShell` installiert. Dateien früherer Versionen, die im Ordner `AWS ToolsPowerShell` vorliegen, werden vom Installationsprogramm automatisch entfernt. Das `PSModulePath` für Windows PowerShell (alle Versionen) wurde in dieser Version aktualisiert und enthält nun den übergeordneten Ordner des Moduls (`AWS ToolsPowerShell`). Bei Systemen mit Windows PowerShell Version 2 wurde die Verknüpfung im Startmenü aktualisiert, sodass das Modul vom neuen Speicherort importiert und anschließend ausgeführt wird`Initialize-AWSDefaults`. Bei Systemen mit Windows PowerShell Version 3 wurde die Verknüpfung im Startmenü aktualisiert,	21. Dezember 2012

sodass der `Import-Module` Befehl entfernt wird, sodass nur noch übrig ist`Initialize-AWSDefaults` . Wenn Sie Ihr PowerShell Profil bearbeitet haben, um eine `Import-Module AWSPowerShell.psd1` Datei auszuführen, müssen Sie es so aktualisieren, dass es auf den neuen Speicherort der Datei verweist (oder, falls Sie PowerShell Version 3 verwenden, entfernen Sie die `Import-Module` Anweisung, da sie nicht mehr benötigt wird). Aufgrund dieser Änderungen wird das PowerShell Modul Tools für Windows jetzt bei der Ausführung als verfügbar es Modul aufgeführt`Get-Module -ListAvailable` . Darüber hinaus wird für Benutzer von Windows PowerShell Version 3 bei der Ausführung eines vom Modul exportierten Cmdlets das Modul automatisch in die aktuelle PowerShell Shell geladen, ohne dass es zuerst verwendet werden muss. `Import-Module` Dadurch wird die interaktive Nutzung der Cmdlets auf einem System mit einer Ausführungsrichtli

	nie aktiviert, die eine Skriptausführung untersagt.	
AWS Tools for Windows PowerShell 1.0.0.0	Erstversion	6. Dezember 2012

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.