

Guida per l'utente

AWS Tools for PowerShell



AWS Tools for PowerShell: Guida per l'utente

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

I marchi e il trade dress di Amazon non possono essere utilizzati in relazione a prodotti o servizi che non siano di Amazon, in qualsiasi modo che possa causare confusione tra i clienti o in qualsiasi modo che denigri o discrediti Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà delle rispettive aziende, che possono o meno essere associate, collegate o sponsorizzate da Amazon.

Table of Contents

Cosa sono gli AWS Tools for PowerShell?	1
Manutenzione e supporto per le versioni principali dell'SDK	2
AWS.Tools	2
AWSPowerShell.NetCore	3
AWSPowerShell	3
Come utilizzare questa guida	4
Installazione	5
Installazione su Windows	5
Prerequisiti	6
Installazione di AWS.Tools	6
Installa AWSPowerShell. NetCore	9
Installa AWSPowerShell	10
Abilitazione dell'esecuzione di uno script	11
Controllo delle versioni	12
Aggiornamento AWS Tools for PowerShell	14
Installazione su Linux o macOS	16
Panoramica della configurazione	16
Prerequisiti	6
Installazione di AWS.Tools	17
Installa AWSPowerShell. NetCore	20
Esecuzione di uno script	11
Configurazione della console PowerShell	22
Inizializza la tua sessione PowerShell	22
Controllo delle versioni	12
Aggiornamento di AWS Tools for PowerShell su Linux o macOS	24
Informazioni correlate	25
Migrazione da AWS Tools for PowerShell versione 3.3 alla versione 4	25
Nuova versione AWS.Tools completamente modularizzata	25
Nuovo cmdlet Get-AWSService	26
Nuovo parametro -Select per controllare l'oggetto restituito da un cmdlet	26
Limitazione più coerente del numero di elementi nell'output	28
Parametri di flusso più facili da usare	29
Estendere la pipe in base al nome della proprietà	29
Parametri comuni statici	30

AWS.Tools Dichiarare e applica i parametri obbligatori	30
Tutti i parametri sono "nullable"	30
Rimozione di funzionalità precedentemente deprecate	31
Nozioni di base	32
Configurazione dell'autenticazione degli strumenti	32
Abilitazione e configurazione di IAM Identity Center	33
Configura gli strumenti per PowerShell utilizzare IAM Identity Center.	33
Avviare una sessione del portale di AWS accesso	35
Esempio	36
Informazioni aggiuntive	37
Usa il AWS CLI	37
Specificare AWS le regioni	41
Specificare un endpoint personalizzato o non standard	43
Informazioni aggiuntive	43
Configurazione dell'identità federata	43
Prerequisiti	44
In che modo un utente con identità federata ottiene l'accesso federato alle API del servizio AWS	44
Come funziona il supporto SAML con gli AWS Tools for PowerShell	46
Come utilizzare la configurazione dei cmdlet SAML PowerShell	47
Ulteriori letture	52
Individuazione di cmdlet e alias	52
Individuare i cmdlet	52
Nomi dei cmdlet e degli alias	59
Indirizzamento e \$AWSHistory	63
\$AWSHistory	64
Risoluzione di credenziali e profili	68
Ordine di ricerca credenziali	68
Utenti e ruoli	69
Utenti e set di autorizzazioni	69
Ruoli di servizio	69
Uso delle credenziali legacy	70
Avvertenze e linee guida importanti	71
AWSCredenziali	72
Credenziali condivise	81
Utilizzo dei servizi AWS	87

Codifica concatenazione dei file PowerShell	87
Oggetti restituiti per gli strumenti PowerShell	88
Amazon EC2	88
Amazon S3	88
AWS Lambda e AWS Tools for PowerShell	89
Amazon SNS e Amazon SQS	89
CloudWatch	89
Vedi anche	89
Argomenti	89
Amazon S3 e Tools for Windows PowerShell	90
Creare un bucket Amazon S3, verificare la sua Regione e rimuoverlo se si desidera	91
Configurare un bucket Amazon S3 come website e abilitare la registrazione	92
Caricare oggetti in un bucket Amazon S3	92
Eliminare bucket e oggetti Amazon S3	94
Caricamento dei contenuti di testo in linea su Amazon S3	96
Amazon EC2 e Tools for Windows PowerShell	96
Crea una coppia di chiavi	97
Crea un gruppo di sicurezza	99
Trovare una AMI	104
Avviare un'istanza	107
AWS Lambda e AWS Tools for PowerShell	112
Prerequisiti	6
Installa il modulo utilizzando AWSLambdaPSCore	113
Vedi anche	89
Amazon SQS, Amazon SNS e Tools for Windows PowerShell	113
Creare una coda Amazon SQS e ottenere l'ARN della coda	114
Creare un argomento Amazon SNS.	114
Concedere le autorizzazioni all'argomento SNS	114
Sottoscrivere la coda all'argomento SNS	115
Concedere le autorizzazioni	115
Verificare i risultati	116
Utilizzo degli AWS Tools for Windows PowerShell per pubblicare in CloudWatch	117
Pubblicare un parametro personalizzato nel pannello di controllo CloudWatch	117
Vedi anche	89
Uso di ClientConfig	118
Uso del parametro ClientConfig	118

Uso di una proprietà non definita	119
Impostazione della Regione AWS	119
Esempi di codice	121
Azioni e scenari	121
ACM	123
AppStream 2.0	128
Aurora	154
Auto Scaling	156
Budget AWS	192
AWS Cloud9	193
AWS CloudFormation	200
CloudFront	210
CloudTrail	218
CloudWatch	224
CodeCommit	228
CodeDeploy	233
CodePipeline	252
Amazon Cognito Identity	270
AWS Config	274
Device Farm	293
AWS Directory Service	294
AWS DMS	320
DynamoDB	321
Amazon EC2	336
Amazon ECR	466
Amazon ECS	468
Amazon EFS	474
Amazon EKS	481
Elastic Load Balancing - Versione 1	493
Elastic Load Balancing - Versione 2	514
Amazon FSx	538
AWS Glue	546
AWS Health	547
IAM	549
Kinesis	622
Lambda	626

Amazon ML	639
Macie	645
AWS OpsWorks	646
Listino prezzi AWS	647
Gruppi di risorse	650
Resource Groups Tagging API	658
Route 53	663
Amazon S3	678
S3 Glacier	713
Amazon SES	717
Amazon SNS	719
Amazon SQS	720
AWS STS	732
AWS Support	737
Systems Manager	743
Amazon Translate	816
AWS WAFV2	817
WorkSpaces	818
Sicurezza	834
Protezione dei dati	834
Crittografia dei dati	835
Identity and Access Management	836
Destinatari	836
Autenticazione con identità	837
Gestione dell'accesso con policy	841
Funzionamento dei Servizi AWS con IAM	843
Risoluzione dei problemi di identità e accesso in AWS	843
Convalida della conformità	845
Applicazione di una versione minima di TLS	847
Ulteriori considerazioni sulla sicurezza	847
Registrazione di informazioni sensibili	847
Riferimenti cmdlet	848
Cronologia dei documenti	849
.....	dcclv

Cosa sono gli AWS Tools for PowerShell?

Gli AWS Tools for PowerShell sono un set di moduli di PowerShell basati sulle funzionalità esposte dall'AWS SDK for .NET. Gli AWS Tools for PowerShell consentono di eseguire lo script di operazioni sulle risorse AWS tramite la riga di comando PowerShell.

I cmdlet forniscono un'esperienza PowerShell idiomatica per specificare i parametri e gestire i risultati, anche se sono implementati utilizzando le varie API di query HTTP del servizio AWS. Ad esempio, i cmdlet per gli AWS Tools for PowerShell supportano l'indirizzamento di PowerShell, per cui è possibile indirizzare gli oggetti PowerShell dentro e fuori dai cmdlet.

Gli AWS Tools for PowerShell offrono flessibilità per quanto riguarda la gestione delle credenziali e il supporto per l'infrastruttura AWS Identity and Access Management (IAM). Puoi utilizzare gli strumenti con credenziali utente IAM, token di sicurezza temporanei e ruoli IAM.

Gli AWS Tools for PowerShell supportano lo stesso set di servizi e regioni AWS che sono supportati dall'SDK. Puoi installare AWS Tools for PowerShell in computer che eseguono i sistemi operativi Windows, Linux o macOS.

Note

AWS Tools for PowerShell versione 4 è l'ultima versione principale ed è un aggiornamento compatibile con le versioni precedenti a AWS Tools for PowerShell versione 3.3. Aggiunge miglioramenti significativi pur mantenendo il comportamento dei cmdlet esistenti. Gli script esistenti dovrebbero continuare a funzionare dopo l'aggiornamento alla nuova versione, ma si consiglia di verificarli accuratamente prima di eseguire l'aggiornamento. Per ulteriori informazioni sulle modifiche nella versione 4, consulta [Migrazione da AWS Tools for PowerShell versione 3.3 alla versione 4](#).

Gli AWS Tools for PowerShell sono disponibili come i seguenti tre pacchetti distinti:

- [AWS.Tools](#)
- [AWSPowerShell.NetCore](#)
- [AWSPowerShell](#)

Manutenzione e supporto per le versioni principali dell'SDK

Per informazioni sulla manutenzione e sul supporto per le versioni principali dell'SDK e le relative dipendenze sottostanti, consulta quanto segue nella [Guida di riferimento degli strumenti e degli SDK AWS](#):

- [Policy di manutenzione degli SDK AWS e degli strumenti](#)
- [Matrice di supporto delle versioni degli SDK AWS e degli strumenti](#)

AWS.Tools: una versione modulare degli AWS Tools for PowerShell

PowerShell Gallery **AWS.Tools**

ZIP Archive **AWS.Tools**

Questa versione di AWS Tools for PowerShell è la versione consigliata per qualsiasi computer che esegue PowerShell in un ambiente di produzione. Poiché è suddiviso in moduli, devi scaricare e caricare solo i moduli per i servizi che desideri utilizzare. Ciò riduce i tempi di download, l'utilizzo della memoria e, nella maggior parte dei casi, consente l'importazione automatica dei cmdlet AWS.Tools con la necessità di chiamare manualmente prima `Import-Module`.

Questa è la versione più recente di AWS Tools for PowerShell e puoi eseguirla su tutti i sistemi operativi supportati, inclusi Windows, Linux e macOS. Questo pacchetto fornisce un modulo di installazione comune, `AWS.Tools.Installer`, un modulo comune, `AWS.Tools.Common` e un modulo per ogni servizio AWS, ad esempio, `AWS.Tools.EC2`, `AWS.Tools.IAM`, `AWS.Tools.S3`, e così via.

Il modulo `AWS.Tools.Installer` fornisce cmdlet che consentono di installare, aggiornare e rimuovere i moduli per ciascuno dei servizi AWS. I cmdlet in questo modulo garantiscono automaticamente di disporre di tutti i moduli dipendenti necessari per supportare i moduli che desideri utilizzare.

Il modulo `AWS.Tools.Common` fornisce cmdlet per la configurazione e l'autenticazione che non sono specifici del servizio. Per utilizzare i cmdlet per un servizio AWS, devi solo eseguire il comando. PowerShell importa automaticamente il modulo `AWS.Tools.Common` e il modulo per il servizio AWS di cui desideri eseguire il cmdlet. Questo modulo viene installato automaticamente se utilizzi il modulo `AWS.Tools.Installer` per installare i moduli di servizio.

Puoi installare questa versione di AWS Tools for PowerShell sui computer che eseguono:

- PowerShell Core 6.0 o versioni successive su Windows, Linux o macOS.
- Windows PowerShell 5.1 o versioni successive su Windows con .NET Framework 4.7.2 o versioni successive.

In questa guida, quando abbiamo bisogno di specificare solo questa versione, vi facciamo riferimento con il nome del modulo: *AWS.Tools*.

AWSPowerShell.NetCore: una versione a modulo singolo degli AWS Tools for PowerShell

PowerShell Gallery **AWSPowerShell.NetCore**

ZIP Archive **AWSPowerShell.NetCore**

Questa versione è costituita da un modulo singolo di grandi dimensioni che contiene il supporto per tutti i servizi AWS. Prima di poter utilizzare questo modulo, devi importarlo manualmente.

Puoi installare questa versione di AWS Tools for PowerShell sui computer che eseguono:

- PowerShell Core 6.0 o versioni successive su Windows, Linux o macOS.
- Windows PowerShell 3.0 o versioni successive su Windows con .NET Framework 4.7.2 o versioni successive.

In questa guida, quando abbiamo bisogno di specificare solo questa versione, vi facciamo riferimento con il nome del modulo: *AWSPowerShell.NetCore*.

AWSPowerShell: una versione a modulo singolo per Windows PowerShell

PowerShell Gallery **AWSPowerShell**

ZIP Archive **AWSPowerShell**

Questa versione degli AWS Tools for PowerShell è compatibile e installabile solo su computer Windows che eseguono Windows PowerShell dalla versione 2.0 alla 5.1. Non è compatibile con

PowerShell Core 6.0 o versioni successive o con qualsiasi altro sistema operativo (Linux o macOS). Questa versione è costituita da un modulo singolo di grandi dimensioni che contiene il supporto per tutti i servizi AWS.

In questa guida, quando abbiamo bisogno di specificare solo questa versione, vi facciamo riferimento con il nome del modulo: `AWSPowerShell`.

Come utilizzare questa guida

La guida è suddivisa nelle seguenti sezioni principali.

[Installazione di AWS Tools for PowerShell](#)

Questa sezione spiega come installare gli AWS Tools for PowerShell. Include istruzioni su come registrarsi ad AWS se non si dispone già di un account e su come creare un utente IAM da utilizzare per eseguire i cmdlet.

[Nozioni di base su AWS Tools for Windows PowerShell](#)

In questa sezione vengono illustrati i concetti fondamentali per utilizzare gli AWS Tools for PowerShell, come specificare le credenziali e le regioni AWS, ricercare i cmdlet per un determinato servizio e utilizzare gli alias per i cmdlet.

[Utilizzo dei servizi AWS in AWS Tools for PowerShell](#)

Questa sezione include informazioni sull'utilizzo degli AWS Tools for PowerShell per eseguire alcune delle più comuni attività AWS.

Installazione di AWS Tools for PowerShell

Per installare e utilizzare correttamente i cmdlet AWS Tools for PowerShell, vedi la procedura descritta nei seguenti argomenti.

Argomenti

- [Installazione di AWS Tools for PowerShell su Windows](#)
- [Installazione AWS Tools for PowerShell su Linux o macOS](#)
- [Migrazione da AWS Tools for PowerShell versione 3.3 alla versione 4](#)

Installazione di AWS Tools for PowerShell su Windows

Un computer basato su Windows può eseguire qualsiasi opzione del pacchetto: AWS Tools for PowerShell

- [AWS.Tools](#)- La versione modulare di AWS Tools for PowerShell. Ogni AWS servizio è supportato da un proprio piccolo modulo individuale, con moduli `AWS.Tools.Common` di supporto condivisi e `AWS.Tools.Installer`.
- [AWSPowerShell.NetCore](#)- La versione a modulo singolo di grandi dimensioni di AWS Tools for PowerShell. Tutti i AWS servizi sono supportati da questo unico modulo di grandi dimensioni.

Note

Tieni presente che il singolo modulo potrebbe essere troppo grande per essere utilizzato con le funzioni [AWS Lambda](#). Utilizza invece la versione modulare mostrata in precedenza.

- [AWSPowerShell](#) - La versione legacy specifica per Windows, con modulo singolo di grandi dimensioni AWS Tools for PowerShell. Tutti i AWS servizi sono supportati da questo unico modulo di grandi dimensioni.

Il pacchetto scelto dipende dalla versione e dall'edizione di Windows in esecuzione.

Note

Gli strumenti per Windows PowerShell (AWSPowerShell modulo) sono installati per impostazione predefinita su tutte le Amazon Machine Images (AMI) basate su Windows.

La configurazione di AWS Tools for PowerShell prevede le seguenti attività di alto livello, descritte in dettaglio in questo argomento.

1. Installa l'opzione di AWS Tools for PowerShell pacchetto appropriata per il tuo ambiente.
2. Verificare che l'esecuzione dello script sia abilitata eseguendo il cmdlet `Get-ExecutionPolicy`.
3. Importa il AWS Tools for PowerShell modulo nella tua PowerShell sessione.

Prerequisiti

Le versioni più recenti di PowerShell, tra cui PowerShell Core, sono disponibili come download da Microsoft all'indirizzo [Installazione di varie versioni del PowerShell](#) sito Web di Microsoft.

Installazione di **AWS.Tools** su Windows

È possibile installare la versione modulare di AWS Tools for PowerShell su computer che eseguono Windows con Windows PowerShell 5.1 o PowerShell Core 6.0 o versione successiva. Per informazioni su come installare PowerShell Core, vedere [Installazione di varie versioni di PowerShell sul sito Web di Microsoft](#).

È possibile installare **AWS.Tools** in uno dei tre modi:

- Utilizzo dei cmdlet nel modulo `AWS.Tools.Installer`. Questo modulo semplifica l'installazione e l'aggiornamento di altri **AWS.Tools** moduli. `AWS.Tools.Installer` richiede `PowerShellGet` e ne scarica e installa automaticamente una versione aggiornata. `AWS.Tools.Installer` mantiene automaticamente sincronizzate le versioni del modulo. Quando si installa o si esegue l'aggiornamento a una versione più recente di un modulo, i cmdlet inclusi aggiornano `AWS.Tools.Installer` automaticamente tutti gli altri **AWS.Tools** moduli alla stessa versione.


Questo metodo è descritto nella procedura che segue.

- Scaricando i moduli da [AWS.Tools.zip](#) ed estraendoli in una delle cartelle dei moduli. È possibile scoprire le cartelle dei moduli visualizzando il valore della variabile di ambiente `PSModulePath`.

- Installazione di ogni modulo di servizio dalla PowerShell Galleria utilizzando il `Install-Module` cmdlet.

Per l'installazione **AWS.Tools** su Windows utilizzando il modulo **AWS.Tools.Installer**

1. Avvia una PowerShell sessione.

 Note

Ti consigliamo di non lavorare PowerShell come amministratore con autorizzazioni elevate, tranne quando richiesto dall'attività in questione. Ciò è dovuto al potenziale rischio per la sicurezza ed è in contrasto con il principio del privilegio minimo.

2. Per installare il pacchetto **AWS.Tools** suddiviso in moduli, esegui il comando seguente.

```
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
```

Se ricevi una notifica in cui è indicato che il repository non è attendibile, ti viene chiesto se desideri installarlo comunque. Inserisci **y** per consentire l' PowerShell installazione del modulo. Per evitare il prompt e installare il modulo senza dover considerare attendibile il repository, puoi eseguire il comando con il parametro `-Force`.

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

3. È ora possibile installare il modulo per ogni AWS servizio che si desidera utilizzare utilizzando il `Install-AWSToolsModule` cmdlet. Ad esempio, il seguente comando installa i moduli Amazon EC2 e Amazon S3. Questo comando installa anche tutti i moduli dipendenti necessari per il funzionamento del modulo specificato. Ad esempio, quando installi il primo modulo di servizio **AWS.Tools**, viene installato anche **AWS.Tools.Common**. Si tratta di un modulo condiviso richiesto da tutti i moduli di AWS servizio. Rimuove anche le versioni precedenti dei moduli e aggiorna altri moduli alla stessa versione più recente.

```
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

Il cmdlet `Install-AWSToolsModule` scarica tutti i moduli richiesti da PSRepository con nome PSGallery (<https://www.powershellgallery.com/>) e lo considera un'origine attendibile. Usa il comando `Get-PSRepository -Name PSGallery` per ulteriori informazioni su PSRepository.

Per impostazione predefinita, il comando precedente installa i moduli nella cartella `%USERPROFILE%\Documents\WindowsPowerShell\Modules`. Per installarlo AWS Tools for PowerShell per tutti gli utenti di un computer, è necessario eseguire il comando seguente in una PowerShell sessione avviata come amministratore. Ad esempio, il seguente comando installa il modulo IAM nella cartella `%ProgramFiles%\WindowsPowerShell\Modules` che è accessibile a tutti gli utenti.

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

Per installare altri moduli, esegui comandi simili con i nomi dei moduli appropriati, come si trova nella [PowerShell Galleria](#).

Installa AWSPowerShell. NetCore su Windows

È possibile installare il AWSPowerShell. NetCore su computer che eseguono Windows con PowerShell versioni da 3 a 5.1 oppure PowerShell Core 6.0 o versioni successive. Per informazioni su come installare PowerShell Core, vedi [Installazione di varie versioni PowerShell](#) sul PowerShell sito Web di Microsoft.

È possibile installare AWSPowerShell. NetCore in due modi

- Scaricando il modulo da [AWSPowerShell. NetCore.zip](#) ed estraendolo in una delle directory del modulo. Puoi individuare le directory dei moduli visualizzando il valore della variabile di ambiente `PSModulePath`.
- Installazione dalla PowerShell Galleria utilizzando il `Install-Module` cmdlet, come descritto nella procedura seguente.

Per installare. AWSPowerShell NetCore dalla PowerShell Galleria utilizzando il cmdlet `Install-Module`

Per installare il. AWSPowerShell NetCore dalla PowerShell Galleria, sul computer deve essere installata la PowerShell versione 5.0 o versione successiva oppure la versione PowerShell 3 o versione successiva. [PowerShellGet](#) Esegui il comando seguente.

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

Se esegui PowerShell come amministratore, il comando precedente viene installato AWS Tools for PowerShell per tutti gli utenti del computer. Se si esegue PowerShell come utente standard senza autorizzazioni di amministratore, lo stesso comando viene installato solo AWS Tools for PowerShell per l'utente corrente.

Per effettuare l'installazione solo per l'utente corrente quando tale utente dispone delle autorizzazioni di amministratore, esegui il comando con il set di parametri `-Scope CurrentUser`, come indicato di seguito.

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


Sebbene la PowerShell versione 3.0 e le versioni successive in genere carichino i moduli nella PowerShell sessione la prima volta che si esegue un cmdlet nel modulo, il AWSPowerShell NetCore il modulo è troppo grande per supportare questa funzionalità. È invece necessario caricare esplicitamente il AWSPowerShell. NetCore Modulo principale nella PowerShell sessione eseguendo il comando seguente.

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

Per caricare il AWSPowerShell. NetCore inserisci automaticamente il modulo in una PowerShell sessione, aggiungi quel comando al tuo PowerShell profilo. Per ulteriori informazioni sulla modifica del PowerShell profilo, consulta [About Profiles](#) nella PowerShell documentazione.

Installa AWSPowerShell su Windows PowerShell

Puoi installarli AWS Tools for Windows PowerShell in uno dei due modi seguenti:

- Scaricando il modulo [AWSPowerShellDa.zip](#) ed estraendolo in una delle directory del modulo. Puoi individuare le directory dei moduli visualizzando il valore della variabile di ambiente `PSModulePath`.
- Installazione dalla PowerShell Galleria utilizzando il `Install-Module` cmdlet come descritto nella procedura seguente.

Per eseguire l'installazione AWSPowerShell dalla PowerShell Galleria utilizzando il cmdlet `Install-Module`

Puoi installarlo AWSPowerShell dalla PowerShell Galleria se esegui la PowerShell versione 5.0 o successiva oppure la versione 3 o una versione successiva. [PowerShellGet](#) PowerShell Puoi installare e aggiornare AWSPowerShell dalla [PowerShellGalleria](#) di Microsoft eseguendo il comando seguente.

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

Per caricare automaticamente il AWSPowerShell modulo in una PowerShell sessione, aggiungi il `import-module` cmdlet precedente al tuo PowerShell profilo. Per ulteriori informazioni sulla modifica del PowerShell profilo, vedere [Informazioni sui profili](#) nella PowerShell documentazione.

Note

Gli strumenti per Windows PowerShell sono installati per impostazione predefinita su tutte le Amazon Machine Images (AMI) basate su Windows.

Abilitazione dell'esecuzione di uno script

Per caricare i AWS Tools for PowerShell moduli, è necessario abilitare PowerShell l'esecuzione degli script. Per abilitare l'esecuzione dello script, eseguire il cmdlet `Set-ExecutionPolicy` per impostare una policy `RemoteSigned`. Per ulteriori informazioni, consulta [Informazioni sulle policy di esecuzione](#) sul sito Web Microsoft Technet.

Note

Questo è un requisito solo per i computer che eseguono Windows. La restrizione di sicurezza `ExecutionPolicy` non è presente su altri sistemi operativi.

Per abilitare l'esecuzione dello script

1. Per impostare la policy di esecuzione sono necessari i diritti di amministratore. Se non hai effettuato l'accesso come utente con diritti di amministratore, apri una PowerShell sessione come amministratore. Scegli Start, quindi seleziona Tutti i programmi. Scegli Accessori, quindi scegli Windows PowerShell. Fate clic con PowerShell il pulsante destro del mouse su Windows e nel menu contestuale scegliete Esegui come amministratore.
2. Nel prompt dei comandi inserisci quanto segue.

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

Note

Su un sistema a 64 bit, è necessario eseguire questa operazione separatamente per la versione a 32 bit di PowerShell Windows PowerShell (x86).

Se la politica di esecuzione non è impostata correttamente, PowerShell mostra il seguente errore ogni volta che si tenta di eseguire uno script, ad esempio il profilo.

```
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
```

Il programma di PowerShell installazione di Tools for Windows aggiorna automaticamente il [PS ModulePath](#) per includere la posizione della directory che contiene il AWSPowerShell modulo.

Poiché PSModulePath include la posizione della directory del AWS modulo, il `Get-Module -ListAvailable` cmdlet mostra il modulo.

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

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

Controllo delle versioni

AWS rilascia AWS Tools for PowerShell periodicamente nuove versioni di per supportare nuovi AWS servizi e funzionalità. Per determinare la versione degli strumenti installata, eseguire il `AWSPowerShellVersion` cmdlet [Get-](#).

```
PS > Get-AWSPowerShellVersion
```

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

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

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

```
This software includes third party software subject to the following copyrights:
```

```
- Logging from log4net, Apache License
[http://logging.apache.org/log4net/license.html]
```

È inoltre possibile aggiungere il `-ListServiceVersionInfo` parametro a un `AWSPowerShellVersion` comando `Get-` per visualizzare un elenco dei AWS servizi supportati nella versione corrente degli strumenti. Se utilizzi l'opzione `AWS.Tools.*` suddivisa in moduli, vengono visualizzati solo i moduli attualmente importati.

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

```
...
```

Service	Noun	Prefix	Module Name	SDK
Assembly				
Version				
-----			-----	-----

Alexa For Business 3.7.0.11	ALXB		AWS.Tools.AlexaForBusiness	
Amplify Backend 3.7.0.11	AMPB		AWS.Tools.AmplifyBackend	
Amazon API Gateway 3.7.0.11	AG		AWS.Tools.APIGateway	
Amazon API Gateway Management API 3.7.0.11	AGM		AWS.Tools.ApiGatewayManagementApi	
Amazon API Gateway V2 3.7.0.11	AG2		AWS.Tools.ApiGatewayV2	
Amazon Appflow 3.7.1.4	AF		AWS.Tools.Appflow	
Amazon Route 53 3.7.0.12	R53		AWS.Tools.Route53	
Amazon Route 53 Domains 3.7.0.11	R53D		AWS.Tools.Route53Domains	
Amazon Route 53 Resolver 3.7.1.5	R53R		AWS.Tools.Route53Resolver	

```
Amazon Simple Storage Service (S3) S3      AWS.Tools.S3
3.7.0.13
...
```

Per determinare la versione in esecuzione, accedi `$PSVersionTable` per visualizzare il contenuto della [variabile VersionTable automatica](#) `$PS`. PowerShell

```
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
```

Aggiornamento di AWS Tools for PowerShell su Windows

Periodicamente, man mano che AWS Tools for PowerShell vengono rilasciate versioni aggiornate di, è necessario aggiornare la versione in esecuzione localmente.

Aggiornate i moduli modulari **AWS.Tools**

Per aggiornare i `AWS.Tools` moduli alla versione più recente, esegui il seguente comando:

```
PS > Update-AWSToolsModule -Cleanup
```

Questo comando aggiorna tutti i moduli `AWS.Tools` attualmente installati e, dopo un aggiornamento riuscito, rimuove altre versioni installate.

Note

Il cmdlet `Update-AWSToolsModule` scarica tutti i moduli dal `PSRepository` con nome `PSGallery` (<https://www.powershellgallery.com/>) e lo considera un'origine attendibile. Usa

il comando `Get-PSRepository -Name PSGallery` per ulteriori informazioni su questo `PSRepository`.

Aggiorna gli strumenti per PowerShell Core

Esegui il `Get-AWSPowerShellVersion` cmdlet per determinare la versione in esecuzione e confrontala con la versione di Tools for Windows PowerShell disponibile sul sito Web [PowerShell Gallery](#). Consigliamo di controllare ogni due o tre settimane. Il supporto per nuovi comandi e AWS servizi è disponibile solo dopo l'aggiornamento a una versione con tale supporto.

Prima di installare una versione più recente di `AWSPowerShell.NetCore`, disinstalla il modulo esistente. Chiudi tutte PowerShell le sessioni aperte prima di disinstallare il pacchetto esistente. Per disinstallare il pacchetto, eseguire il comando seguente.

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

Dopo aver disinstallato il pacchetto, installa il modulo aggiornato eseguendo il comando riportato qui di seguito.

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

Dopo l'installazione, esegui il comando `Import-Module AWSPowerShell.NetCore` per caricare i cmdlet aggiornati nella sessione PowerShell .

Aggiorna gli strumenti per Windows PowerShell

Esegui il `Get-AWSPowerShellVersion` cmdlet per determinare la versione in esecuzione e confrontala con la versione di Tools for Windows PowerShell disponibile sul sito Web [PowerShell Gallery](#). Consigliamo di controllare ogni due o tre settimane. Il supporto per nuovi comandi e AWS servizi è disponibile solo dopo l'aggiornamento a una versione con tale supporto.

- Se hai effettuato l'installazione utilizzando il cmdlet `Install-Module`, esegui il comando riportato qui di seguito.

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

- Se hai effettuato l'installazione utilizzando un file ZIP scaricato:

1. Scarica la versione più recente dal sito PowerShell Web [Tools for](#). Confronta il numero di versione del pacchetto nel nome del file scaricato con il numero di versione visualizzato quando esegui il cmdlet `Get-AWSPowerShellVersion`.
2. Se il numero della versione scaricata è superiore a quello della versione installata, chiudi tutti gli strumenti per le PowerShell console Windows.
3. Installa la versione più recente degli strumenti per Windows. PowerShell

Dopo l'installazione, `Import-Module AWSPowerShell` esegui per caricare i cmdlet aggiornati nella sessione. PowerShell Oppure esegui la AWS Tools for PowerShell console personalizzata dal menu Start.

Installazione AWS Tools for PowerShell su Linux o macOS

Questo argomento fornisce istruzioni su come installarlo AWS Tools for PowerShell su Linux o macOS.

Panoramica della configurazione

Per l'installazione AWS Tools for PowerShell su un computer Linux o macOS, puoi scegliere tra due opzioni di pacchetto:

- [AWS.Tools](#)— La versione modulare di. AWS Tools for PowerShell Ogni AWS servizio è supportato da un proprio piccolo modulo individuale, con moduli di supporto condivisi. `AWS.Tools.Common`
- [AWSPowerShell.NetCore](#)— La versione a modulo singolo di grandi dimensioni di AWS Tools for PowerShell. Tutti i AWS servizi sono supportati da questo unico modulo di grandi dimensioni.

Note

Tieni presente che il singolo modulo potrebbe essere troppo grande per essere utilizzato con le funzioni [AWS Lambda](#). Utilizza invece la versione modulare mostrata in precedenza.

L'impostazione di una di queste versioni su un computer su cui è in esecuzione Linux o macOS comporta le seguenti attività, descritte in dettaglio più avanti in questo argomento:

1. Installa PowerShell Core 6.0 o versione successiva su un sistema supportato.

2. Dopo aver installato PowerShell Core, inizia PowerShell eseguendolo `pwsh` nella shell del sistema.
3. Installa uno `AWS.Tools` o `AWSPowerShell`. `NetCore`.
4. Esegui il `Import-Module` cmdlet appropriato per importare il modulo nella sessione PowerShell.
5. Esegui il `AWSDefaultConfiguration` cmdlet [Initialize- per fornire le](#) tue credenziali. AWS

Prerequisiti

Per eseguire AWS Tools for PowerShell Core, sul computer deve essere in esecuzione PowerShell Core 6.0 o versione successiva.

- Per un elenco delle versioni supportate della piattaforma Linux e per informazioni su come installare la versione più recente di PowerShell su un computer basato su Linux, vedi [Installazione PowerShell su Linux sul sito Web](#) di Microsoft. Alcuni sistemi operativi basati su Linux, ad esempio Arch, Kali e Raspbian non sono supportati ufficialmente, ma hanno vari livelli di supporto dalla comunità.
- Per informazioni sulle versioni di macOS supportate e su come installare la versione più recente di PowerShell macOS, consulta [Installazione su PowerShell macOS sul sito Web](#) di Microsoft.

Installazione di **AWS.Tools** su Linux o macOS

È possibile installare la versione modulare di AWS Tools for PowerShell su computer che PowerShell eseguono Core 6.0 o versione successiva. Per informazioni su come installare PowerShell Core, vedi [Installazione di varie versioni PowerShell](#) sul PowerShell sito Web di Microsoft.

È possibile installare `AWS.Tools` in uno dei tre modi:

- Utilizzo dei cmdlet nel modulo `AWS.Tools.Installer`. Questo modulo semplifica l'installazione e l'aggiornamento di altri `AWS.Tools` moduli. `AWS.Tools.Installer` richiede `PowerShellGet` e ne scarica e installa automaticamente una versione aggiornata. `AWS.Tools.Installer` mantiene automaticamente sincronizzate le versioni del modulo. Quando si installa o si esegue l'aggiornamento a una versione più recente di un modulo, i cmdlet inclusi aggiornano `AWS.Tools.Installer` automaticamente tutti gli altri `AWS.Tools` moduli alla stessa versione.

Questo metodo è descritto nella procedura che segue.

- Scaricando i moduli da [AWS.Tools.zip](#) ed estraendoli in una delle directory dei moduli. È possibile scoprire le directory dei moduli stampando il valore della variabile `$Env:PSModulePath`.

- Installazione di ogni modulo di servizio dalla PowerShell Galleria utilizzando il `Install-Module` cmdlet.

Per installare **AWS.Tools** su Linux o macOS utilizzando il modulo **AWS.Tools.Installer**

1. Avvia una sessione PowerShell Core eseguendo il comando seguente.

```
$ pwsh
```

Note

Ti consigliamo di non lavorare PowerShell come amministratore con autorizzazioni elevate, tranne quando richiesto dall'attività in questione. Ciò è dovuto al potenziale rischio per la sicurezza ed è in contrasto con il principio del privilegio minimo.

2. Per installare il pacchetto `AWS.Tools` modularizzato utilizzando il modulo `AWS.Tools.Installer`, eseguire il seguente comando.

```
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
```

Se ricevi una notifica in cui è indicato che il repository non è attendibile, ti viene chiesto se vuoi installarlo comunque. Inserisci **y** per consentire l' PowerShell installazione del modulo. Per evitare il prompt e installare il modulo senza dover considerare attendibile il repository, puoi eseguire il seguente comando.

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

3. Ora puoi installare il modulo per ogni servizio che desideri utilizzare. Ad esempio, il seguente comando installa i moduli Amazon EC2 e Amazon S3. Questo comando installa anche tutti i moduli dipendenti necessari per il funzionamento del modulo specificato. Ad esempio, quando

installi il primo modulo di servizio `AWS.Tools`, viene installato anche `AWS.Tools.Common`. Si tratta di un modulo condiviso richiesto da tutti i moduli AWS di servizio. Rimuove anche le versioni precedenti dei moduli e aggiorna altri moduli alla stessa versione più recente.

```
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

Il cmdlet `Install-AWSToolsModule` scarica tutti i moduli richiesti da `PSRepository` con nome `PSGallery` (<https://www.powershellgallery.com/>) e considera il repository come un'origine attendibile. Usa il comando `Get-PSRepository -Name PSGallery` per ulteriori informazioni su `PSRepository`.

Il comando precedente installa i moduli nelle directory predefinite del sistema. Le directory effettive dipendono dalla distribuzione e dalla versione del sistema operativo e dalla versione installata. PowerShell Ad esempio, se avete installato PowerShell 7 su un sistema simile a RHEL, molto probabilmente i moduli predefiniti si trovano in `/opt/microsoft/powershell/7/Modules` (or `$PSHOME/Modules`) e i moduli utente si trovano molto probabilmente in `~/.local/share/powershell/Modules` Per ulteriori informazioni, vedere

[Installazione PowerShell su Linux](#) sul PowerShell sito Web Microsoft. Per vedere dove sono installati i moduli, esegui il comando seguente:

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

Per installare altri moduli, esegui comandi simili con i nomi dei moduli appropriati, come si trova nella [PowerShell Galleria](#).

Installa AWSPowerShell. NetCore su Linux o macOS

Per eseguire l'aggiornamento a una versione più recente di AWSPowerShell. NetCore, segui le istruzioni riportate in [Aggiornamento di AWS Tools for PowerShell su Linux o macOS](#). Disinstalla le versioni precedenti di AWSPowerShell. NetCore prima.

È possibile installare AWSPowerShell. NetCore in due modi:

- Scaricando il modulo da [AWSPowerShell.NetCore.zip](#) ed estraendolo in una delle directory del modulo. È possibile scoprire le directory dei moduli stampando il valore della variabile `$Env:PSModulePath`.
- Installazione dalla PowerShell Galleria utilizzando il `Install-Module` cmdlet come descritto nella procedura seguente.

Per installare AWSPowerShell NetCore su Linux o macOS utilizzando il cmdlet `Install-Module`

Avviare una sessione PowerShell Core eseguendo il comando seguente.

```
$ pwsh
```

Note

Ti consigliamo di non iniziare PowerShell eseguendo `sudo pwsh` l'esecuzione PowerShell con diritti di amministratore elevati. Ciò è dovuto al potenziale rischio per la sicurezza ed è in contrasto con il principio del privilegio minimo.

Per installare il AWSPowerShell NetCore pacchetto a modulo singolo dalla PowerShell Galleria, esegui il seguente comando.

```
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
```

Se ricevi una notifica in cui è indicato che il repository non è attendibile, ti viene chiesto se vuoi installarlo comunque. Inserisci **y** per consentire PowerShell l'installazione del modulo. Per evitare il prompt senza dover considerare attendibile il repository, puoi eseguire il comando riportato qui di seguito.

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

Non è necessario eseguire questo comando come root, a meno che non si desideri installarlo AWS Tools for PowerShell per tutti gli utenti di un computer. Per fare ciò, esegui il comando seguente in una PowerShell sessione con cui hai iniziato `sudo pwsh`.

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

Esecuzione di uno script

Il comando `Set-ExecutionPolicy` non è disponibile sui sistemi non Windows. Puoi eseguire `Get-ExecutionPolicy`, il che dimostra che l'impostazione predefinita dei criteri di esecuzione in PowerShell Core in esecuzione su sistemi non Windows è `Unrestricted`. Per ulteriori informazioni, consulta [Informazioni sulle policy di esecuzione](#) sul sito Web Microsoft Technet.

Poiché `PSModulePath` include la posizione della directory del AWS modulo, il `Get-Module -ListAvailable` cmdlet mostra il modulo installato.

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	

Configurare una PowerShell console per utilizzare (. AWS Tools for PowerShell Core AWSPowerShell NetCore Solo)

PowerShell Core in genere carica automaticamente i moduli ogni volta che si esegue un cmdlet nel modulo. Ma questo non funziona per AWSPowerShell NetCore a causa delle sue grandi dimensioni. Per iniziare a correre AWSPowerShell.NetCore cmdlet, è necessario innanzitutto eseguire il `Import-Module AWSPowerShell.NetCore` comando. Questo non è necessario per i cmdlet nei moduli `AWS.Tools`.

Inizializza la tua sessione PowerShell

Quando si avvia PowerShell su un sistema basato su Linux o macOS dopo aver installato il AWS Tools for PowerShell, è necessario eseguire [Initialize-AWSDefaultConfiguration](#) per specificare quale chiave di accesso utilizzare. AWS Per ulteriori informazioni su `Initialize-AWSDefaultConfiguration`, consulta [Uso delle credenziali AWS](#).

Note

Nelle versioni precedenti (precedenti alla 3.3.96.0) di, questo cmdlet veniva denominato `AWS Tools for PowerShellInitialize-AWSDefaults`

Controllo delle versioni

AWS rilascia AWS Tools for PowerShell periodicamente nuove versioni per supportare nuovi servizi e funzionalità. AWS Per determinare la versione di AWS Tools for PowerShell che è stata installata, eseguire il `AWSPowerShellVersion` cmdlet [Get-](#).

```
PS > Get-AWSPowerShellVersion
```

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

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

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

```
This software includes third party software subject to the following copyrights:
```

```
- Logging from log4net, Apache License  
[http://logging.apache.org/log4net/license.html]
```

Per visualizzare un elenco dei AWS servizi supportati nella versione corrente degli strumenti, aggiungere il `-ListServiceVersionInfo` parametro a un cmdlet [Get-AWSPowerShellVersion](#).

Per determinare la versione in esecuzione, immettere `$PSVersionTable` per visualizzare il contenuto della variabile `$PSVersionTable` [automatica](#). PowerShell

```
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

Aggiornamento di AWS Tools for PowerShell su Linux o macOS

Periodicamente, man mano che AWS Tools for PowerShell vengono rilasciate versioni aggiornate di, è necessario aggiornare la versione in esecuzione localmente.

Aggiorna i moduli modularizzati **AWS.Tools**

Per aggiornare i **AWS.Tools** moduli alla versione più recente, esegui il seguente comando:

```
PS > Update-AWSToolsModule -Cleanup
```

Questo comando aggiorna tutti i moduli **AWS.Tools** attualmente installati e, per i moduli che sono stati aggiornati correttamente, rimuove le versioni precedenti.

Note

Il cmdlet `Update-AWSToolsModule` scarica tutti i moduli dal `PSRepository` con nome `PSGallery` (<https://www.powershellgallery.com/>) e lo considera un'origine attendibile. Usa il comando `Get-PSRepository -Name PSGallery` per ulteriori informazioni su `PSRepository`.

Aggiorna gli strumenti per PowerShell Core

Esegui il `Get-AWSPowerShellVersion` cmdlet per determinare la versione in esecuzione e confrontala con la versione di Tools for Windows PowerShell disponibile sul sito Web [PowerShell Gallery](#). Consigliamo di controllare ogni due o tre settimane. Il supporto per nuovi comandi e AWS servizi è disponibile solo dopo l'aggiornamento a una versione con tale supporto.

Prima di installare una versione più recente di `AWSPowerShell`. `NetCore`, disinstalla il modulo esistente. Chiudi tutte PowerShell le sessioni aperte prima di disinstallare il pacchetto esistente. Per disinstallare il pacchetto, eseguire il comando seguente.

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

Dopo aver disinstallato il pacchetto, installa il modulo aggiornato eseguendo il comando riportato qui di seguito.

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

Dopo l'installazione, esegui il comando `Import-Module AWSPowerShell.NetCore` per caricare i cmdlet aggiornati nella sessione PowerShell .

Informazioni correlate

- [Nozioni di base su AWS Tools for Windows PowerShell](#)
- [Utilizzo dei servizi AWS in AWS Tools for PowerShell](#)

Migrazione da AWS Tools for PowerShell versione 3.3 alla versione 4

AWS Tools for PowerShell versione 4 è un aggiornamento compatibile con le versioni precedenti ad AWS Tools for PowerShell versione 3.3. Aggiunge miglioramenti significativi pur mantenendo il comportamento dei cmdlet esistenti.

Gli script esistenti dovrebbero continuare a funzionare dopo l'aggiornamento alla nuova versione, ma si consiglia di verificarli accuratamente prima di eseguire l'aggiornamento agli ambienti di produzione.

Questa sezione descrive le modifiche e spiega come potrebbero influire sugli script.

Nuova versione **AWS.Tools** completamente modularizzata

La `AWSPowerShell.NetCore` e `AWSPowerShell` i pacchetti erano «monolitici». Ciò significava che tutti i servizi AWS erano supportati nello stesso modulo, rendendolo molto grande e crescendo con l'aggiunta di ogni nuovo servizio AWS e funzionalità. Il nuovo pacchetto `AWS.Tools` è suddiviso in moduli più piccoli che offrono la flessibilità di scaricare e installare solo quelli necessari per i servizi AWS utilizzati. Il pacchetto include un modulo `AWS.Tools.Common` condiviso richiesto da tutti gli altri moduli e un modulo `AWS.Tools.Installer` che semplifica l'installazione, l'aggiornamento e la rimozione dei moduli secondo necessità.

Ciò consente anche l'importazione automatica dei cmdlet alla prima chiamata, senza dover effettuare la prima chiamata `Import-Module`. Tuttavia, per interagire con gli oggetti .NET associati prima di chiamare un cmdlet, è comunque necessario chiamare `Import-Module` per comunicare i PowerShell tipi .NET pertinenti.

Ad esempio, il comando seguente ha un riferimento a `Amazon.EC2.Model.Filter`. Questo tipo di riferimento non può attivare l'importazione automatica, quindi è necessario chiamare prima `Import-Module` o il comando `fallisce`.

```
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
```

Nuovo cmdlet `Get-AWSService`

Per individuare i nomi dei moduli per ogni servizio AWS nella raccolta di moduli `AWS.Tools`, è possibile utilizzare il cmdlet `Get-AWSService`.

```
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
...
```

Nuovo parametro `-Select` per controllare l'oggetto restituito da un cmdlet

La maggior parte dei cmdlet nella versione 4 supporta un nuovo parametro `-Select`. Ogni cmdlet chiama le API del servizio AWS per l'utente utilizzando il metodo AWS SDK for .NET. Il AWS Tools for PowerShell client converte quindi la risposta in un oggetto che è possibile utilizzare negli PowerShell script e reindirizzare ad altri comandi. A volte l'PowerShell oggetto finale ha più campi o proprietà nella risposta originale del necessario e altre volte potresti volere che l'oggetto includa

campi o proprietà della risposta che non sono presenti per impostazione predefinita. Il parametro `-Select` consente di specificare gli elementi inclusi nell'oggetto.NET restituito dal cmdlet.

Ad esempio, il cmdlet [Get-S3Object](#) richiama l'operazione Amazon S3 SDK. [ListObjects](#) Tale [ListObjectsResponse](#) operazione restituisce un oggetto. Tuttavia, per impostazione predefinita, il `Get-S3Object` cmdlet restituisce all'utente solo l'`S3Object` elemento della risposta SDK. PowerShell Nell'esempio seguente, tale oggetto è un array con due elementi.

```
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 versione 4, è possibile specificare `-Select *` per restituire l'oggetto di risposta.NET completo restituito dalla chiamata API SDK.

```
PS > Get-S3Object -BucketName mybucket -Select *

IsTruncated      : False
NextMarker       :
S3Objects        : {file1.txt, file2.txt}
Name             : mybucket
Prefix          :
MaxKeys          : 1000
CommonPrefixes  : {}
Delimiter        :
```

È inoltre possibile specificare il percorso della proprietà nidificata specifica desiderata. L'esempio seguente restituisce solo la proprietà `Key` di ciascun elemento della matrice `S3Objects`.

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

In alcune situazioni può essere utile restituire un parametro cmdlet. Tale operazione può essere eseguita con `-Select ^ParameterName`. Questa funzione sostituisce il parametro `-PassThru`, che è ancora disponibile ma deprecato.

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

[L'argomento di riferimento](#) per ogni cmdlet consente di identificare se supporta il parametro `-Select`.

Limitazione più coerente del numero di elementi nell'output

Le versioni precedenti degli AWS Tools for PowerShell abilitavano l'utilizzo del parametro `-MaxItems` per specificare il numero massimo di oggetti restituiti nell'output finale.

Questo comportamento è stato rimosso da `AWS.Tools`.

Questo comportamento è obsoleto in `AWSPowerShell NetCore` e `AWSPowerShell`, e verranno rimossi da tali versioni in una versione futura.

Se l'API del servizio sottostante supporta un parametro `MaxItems`, è ancora disponibile e funziona come specificato dall'API. Ma non ha più il comportamento ulteriore di limitare il numero di elementi restituiti nell'output del cmdlet.

Per limitare il numero di elementi restituiti nell'output finale, reindirizzare l'output al cmdlet `Select-Object` e specificare il parametro `-First n`, dove *n* è il numero massimo di elementi da includere nell'output finale.

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

Non tutti i servizi AWS supportavano `-MaxItems` allo stesso modo, quindi questa modifica rimuove tale incoerenza e i risultati imprevisti che a volte si sono verificati. Inoltre, `-MaxItems` in combinazione con il nuovo parametro [-Select](#) a volte potrebbe portare a risultati confusi.

Parametri di flusso più facili da usare

Parametri di tipo `Stream` o `byte[]` possono ora accettare valori `string`, `string[]` o `FileInfo`.

Ad esempio, è possibile utilizzare uno dei seguenti esempi.

```
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 converte tutte le stringhe in `byte[]` usando la codifica UTF-8.

Estendere la pipe in base al nome della proprietà

Per rendere l'esperienza utente più coerente, è ora possibile passare l'input della pipeline specificando il nome della proprietà per qualsiasi parametro.

Nell'esempio seguente viene creato un oggetto personalizzato con proprietà con nomi corrispondenti ai nomi dei parametri del cmdlet di destinazione. Quando viene eseguito il cmdlet, vengono automaticamente consumate tali proprietà come parametri.

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

Note

Alcune proprietà supportavano ciò nelle versioni precedenti degli AWS Tools for PowerShell. La versione 4 rende questo più coerente abilitandolo per tutti i parametri.

Parametri comuni statici

Per migliorare la coerenza nella versione 4.0 degli AWS Tools for PowerShell, tutti i parametri sono statici.

Nelle versioni precedenti degli AWS Tools for PowerShell, alcuni parametri comuni come `AccessKey`, `SecretKey`, `ProfileName` o `Region` erano [dinamici](#), mentre tutti gli altri parametri erano statici. Ciò potrebbe creare problemi perché PowerShell associa i parametri statici prima di quelli dinamici. Ad esempio, supponiamo che tu abbia eseguito il seguente comando.

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

Le versioni precedenti di PowerShell associavano il valore `us-west-2` al parametro `-RegionName` statico anziché al parametro `-Region` dinamico. Probabilmente, questo potrebbe confondere gli utenti.

AWS.Tools Dichiarare e applica i parametri obbligatori

I moduli `AWS.Tools.*` ora dichiarano e applicano parametri cmdlet obbligatori. Quando un AWS servizio dichiara che è necessario un parametro di un'API, PowerShell richiede il parametro del cmdlet corrispondente se non lo si è specificato. Questo vale solo per `AWS.Tools`. Per garantire la compatibilità con le versioni precedenti, questo non si applica a `AWSPowerShell NetCore` oppure `AWSPowerShell`.

Tutti i parametri sono “nullable”

È ora possibile assegnare `$null` ai parametri del tipo di valore (numeri e date). Questa modifica non dovrebbe influire sugli script esistenti. Ciò consente di ignorare la richiesta di un parametro obbligatorio. I parametri obbligatori sono applicati solo in `AWS.Tools`.

Se viene eseguito l'esempio seguente utilizzando la versione 4, il medesimo ignora effettivamente la convalida lato client perché si fornisce un «valore» per ogni parametro obbligatorio. Tuttavia, la chiamata API Amazon EC2 fallisce perché il servizio AWS ancora richiede tali informazioni.

```
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
```

Rimozione di funzionalità precedentemente deprecate

Le seguenti funzionalità sono state deprecate nelle versioni precedenti degli AWS Tools for PowerShell e vengono rimosse nella versione 4:

- Rimosso il parametro `-Terminate` dal cmdlet `Stop-EC2Instance`. Usare invece `Remove-EC2Instance`.
- È stato rimosso il `-ProfileName` parametro dal `AWSCredential cmdlet Clear-`. Usare invece `Remove-AWSCredentialProfile`.
- Rimossi i cmdlet `Import-EC2Instance` e `Import-EC2Volume`.

Nozioni di base su AWS Tools for Windows PowerShell

Alcuni argomenti di questa sezione descrivono le nozioni di base relative all'utilizzo degli strumenti per Windows PowerShell dopo aver [installato gli strumenti](#). Ad esempio, illustrano come specificare le [credenziali](#) e la [regione AWS](#) che i gli strumenti per Windows PowerShell devono usare durante l'interazione con AWS.

Gli altri argomenti di questa sezione forniscono informazioni sui modi avanzati di configurazione degli strumenti, dell'ambiente e dei progetti.

Argomenti

- [Configura l'autenticazione dello strumento con AWS](#)
- [Specificare AWS le regioni](#)
- [Configurazione dell'identità federata con AWS Tools for PowerShell](#)
- [Individuazione di cmdlet e alias](#)
- [Indirizzamento e \\$AWSHistory](#)
- [Risoluzione di credenziali e profili](#)
- [Ulteriori informazioni su utenti e ruoli](#)
- [Uso delle credenziali legacy](#)

Configura l'autenticazione dello strumento con AWS

È necessario stabilire in che modo il codice si autentica AWS durante lo sviluppo con. Servizi AWS Esistono diversi modi in cui è possibile configurare l'accesso programmatico alle AWS risorse, a seconda dell'ambiente e dell' AWS accesso a disposizione.

Per visualizzare i vari metodi di autenticazione per Tools for PowerShell, consulta [Autenticazione e accesso nella Guida](#) di riferimento agli AWS SDK e agli strumenti.

Questo argomento presuppone che un nuovo utente si stia sviluppando localmente, non abbia ricevuto un metodo di autenticazione dal datore di lavoro e che lo utilizzerà AWS IAM Identity Center per ottenere credenziali temporanee. Se l'ambiente in uso non rientra in questi presupposti, alcune delle informazioni contenute in questo argomento potrebbero non riguardarti l'utente o potrebbero esserti già state fornite.

La configurazione di questo ambiente richiede diversi passaggi, riassunti di seguito:

1. [Abilitazione e configurazione di IAM Identity Center](#)
2. [Configura gli strumenti per PowerShell utilizzare IAM Identity Center.](#)
3. [Avviare una sessione del portale di AWS accesso](#)

Abilitazione e configurazione di IAM Identity Center

Per utilizzarlo AWS IAM Identity Center, deve prima essere abilitato e configurato. Per maggiori dettagli su come eseguire questa operazione PowerShell, consulta la Fase 1 dell'argomento relativo all'[autenticazione di IAM Identity Center](#) nella Guida di riferimento agli AWS SDK e agli strumenti. In particolare, segui le istruzioni necessarie riportate nella sezione I do not have established access through IAM Identity Center (Non ho stabilito l'accesso tramite IAM Identity Center).

Configura gli strumenti per PowerShell utilizzare IAM Identity Center.

Note

A partire dalla versione 4.1.538 di Tools for PowerShell, il metodo consigliato per configurare le credenziali SSO e avviare una sessione del portale di AWS accesso consiste nell'utilizzare i [Invoke-AWSSSOLogin](#) cmdlet [Initialize-AWSSSOConfiguration](#) and, come descritto in questo argomento. Se non si ha accesso a quella versione degli strumenti per PowerShell (o successiva) o non è possibile utilizzare tali cmdlet, è comunque possibile eseguire queste attività utilizzando il. AWS CLI Per scoprire come, vedi. [Usa il AWS CLI per accedere al portale](#)

La procedura seguente aggiorna il AWS config file condiviso con le informazioni SSO che Tools for PowerShell utilizza per ottenere credenziali temporanee. Come conseguenza di questa procedura, viene avviata anche una sessione del portale di AWS accesso. Se il config file condiviso contiene già informazioni SSO e desideri semplicemente sapere come avviare una sessione del portale di accesso utilizzando gli Strumenti per PowerShell, consulta la sezione successiva di questo argomento, [Avviare una sessione del portale di AWS accesso](#).

1. Se non l'hai ancora fatto, apri PowerShell e installa il file appropriato per AWS Tools for PowerShell il tuo sistema operativo e ambiente, inclusi i cmdlet comuni. Per informazioni su come eseguire questa attività, consultare [Installazione di AWS Tools for PowerShell](#).

Ad esempio, se installi la versione modulare di Tools for PowerShell on Windows, molto probabilmente eseguirai comandi simili ai seguenti:

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

2. Esegui il comando seguente. Sostituisci i valori delle proprietà di esempio con i valori della configurazione di IAM Identity Center. Per informazioni su queste proprietà e su come trovarle, consulta le [impostazioni del provider di credenziali IAM Identity Center](#) nella Guida di riferimento agli AWS SDK e agli strumenti.

```
$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
```

In alternativa, è possibile utilizzare semplicemente il cmdlet da solo e la finestra Initialize-AWSSSOConfiguration Tools for PowerShell richiede i valori delle proprietà.

Considerazioni per determinati valori di proprietà:

- Se hai semplicemente seguito le istruzioni per [abilitare e configurare IAM Identity Center](#), il valore per `-RoleName` potrebbe essere `PowerUserAccess`. Ma se hai creato un set di autorizzazioni IAM Identity Center appositamente per PowerShell il lavoro, utilizzalo invece.
 - Assicurati di utilizzare il Regione AWS punto in cui hai configurato IAM Identity Center.
3. A questo punto, il AWS config file condiviso contiene un profilo chiamato `my-sso-profile` con un set di valori di configurazione a cui è possibile fare riferimento negli Strumenti per PowerShell. Per trovare la posizione di questo file, consulta l'argomento relativo alla [posizione dei file condivisi](#) nella Guida di riferimento per SDK e strumenti AWS .

Tools for PowerShell utilizza il provider di token SSO del profilo per acquisire le credenziali prima di inviare richieste a. AWS Il `sso_role_name` valore, che è un ruolo IAM connesso a un set di

autorizzazioni IAM Identity Center, dovrebbe consentire l'accesso ai dati Servizi AWS utilizzati nell'applicazione.

L'esempio seguente mostra il profilo che è stato creato utilizzando il comando mostrato sopra. Alcuni valori delle proprietà e il relativo ordine potrebbero essere diversi nel profilo attuale. La `sso-session` proprietà del profilo si riferisce alla sezione denominata `my-sso-session`, che contiene le impostazioni per avviare una sessione del portale di AWS accesso.

```
[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. Se hai già una sessione attiva del portale di AWS accesso, la sezione Strumenti ti PowerShell informa che hai già effettuato l'accesso.

In caso contrario, la sezione Strumenti PowerShell tenta di aprire automaticamente la pagina di autorizzazione SSO nel browser Web predefinito. Segui le istruzioni del tuo browser, che potrebbero includere un codice di autorizzazione SSO, nome utente e password e l'autorizzazione ad accedere agli AWS IAM Identity Center account e ai set di autorizzazioni.

Gli strumenti per PowerShell informarti che l'accesso SSO è avvenuto con successo.

Avviare una sessione del portale di AWS accesso

Prima di eseguire i comandi di accesso Servizi AWS, è necessaria una sessione attiva del portale di AWS accesso in modo che Tools for PowerShell possa utilizzare l'autenticazione IAM Identity Center per risolvere le credenziali. Per accedere al portale di AWS accesso, esegui il seguente comando in PowerShell, `-ProfileName my-sso-profile` dov'è il nome del profilo che è stato creato nel `config` file condiviso quando hai seguito la procedura nella sezione precedente di questo argomento.

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

Se hai già una sessione attiva del portale di AWS accesso, il menu Strumenti ti PowerShell informa che hai già effettuato l'accesso.

In caso contrario, la sezione Strumenti PowerShell tenta di aprire automaticamente la pagina di autorizzazione SSO nel browser Web predefinito. Segui le istruzioni del tuo browser, che potrebbero includere un codice di autorizzazione SSO, nome utente e password e l'autorizzazione ad accedere agli AWS IAM Identity Center account e ai set di autorizzazioni.

Gli strumenti per PowerShell informarti che l'accesso SSO è avvenuto con successo.

Per verificare se hai già una sessione attiva, esegui il comando seguente dopo aver installato o importato il `AWS.Tools.SecurityToken` modulo, se necessario.

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

La risposta al `Get-STSCallerIdentity` cmdlet riporta l'account IAM Identity Center e il set di autorizzazioni configurati nel file condiviso. `config`

Esempio

Di seguito è riportato un esempio di come utilizzare IAM Identity Center con gli strumenti per PowerShell. Si presume quanto segue:

- Hai abilitato IAM Identity Center e lo hai configurato come descritto in questo argomento. Le proprietà SSO si trovano nel `my-sso-profile` profilo, che è stato configurato in precedenza in questo argomento.
- Quando accedi tramite i `Invoke-AWSSSOLogin` cmdlet `Initialize-AWSSSOConfiguration` or, l'utente dispone almeno delle autorizzazioni di sola lettura per Amazon S3.
- Alcuni bucket S3 sono disponibili per la visualizzazione da parte dell'utente.

Installa o importa il `AWS.Tools.S3` modulo secondo necessità, quindi utilizza il PowerShell comando seguente per visualizzare un elenco dei bucket S3.

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

Informazioni aggiuntive

- Per ulteriori opzioni sull'autenticazione per gli Strumenti per PowerShell, come l'uso di profili e variabili di ambiente, consulta il capitolo sulla [configurazione](#) nella Guida di riferimento agli AWS SDK e agli strumenti.
- Alcuni comandi richiedono la specificazione di una AWS regione. Esistono diversi modi per farlo, tra cui l'opzione del `-Region` cmdlet, il `[default]` profilo e la variabile di `AWS_REGION` ambiente. Per ulteriori informazioni, consulta [Specificare AWS le regioni](#) questa guida e [l'AWS area geografica nella](#) Guida di riferimento agli AWS SDK e agli strumenti.
- Per ulteriori informazioni sulle best practice, consulta [Best practice per la sicurezza in IAM](#) nella Guida per l'utente di IAM.
- Per creare AWS credenziali a breve termine, consulta [Temporary Security Credentials](#) nella IAM User Guide.
- Per ulteriori informazioni su altri provider di credenziali, consulta l'argomento relativo ai [provider di credenziali standard](#) nella Guida di riferimento per SDK e strumenti AWS .

Argomenti

- [Usa il AWS CLI per accedere al portale](#)

Usa il AWS CLI per accedere al portale

A partire dalla versione 4.1.538 di Tools for PowerShell, il metodo consigliato per configurare le credenziali SSO e avviare una sessione del portale di AWS accesso consiste nell'utilizzare i [Invoke-AWSSSOLogin](#) cmdlet [Initialize-AWSSSOConfiguration](#) and, come descritto in [Configura l'autenticazione dello strumento con AWS](#). Se non si ha accesso a quella versione di Tools for PowerShell (o successiva) o non è possibile utilizzare tali cmdlet, è comunque possibile eseguire queste attività utilizzando il AWS CLI

Configura gli strumenti per PowerShell utilizzare IAM Identity Center tramite. AWS CLI

Se non l'hai già fatto, assicurati di [abilitare e configurare IAM Identity Center](#) prima di procedere.

Le informazioni su come configurare gli strumenti PowerShell per utilizzare IAM Identity Center tramite IAM AWS CLI si trovano nella Fase 2 dell'argomento relativo all'[autenticazione di IAM Identity Center](#) nella Guida di riferimento agli AWS SDK e agli strumenti. Dopo aver completato questa configurazione, il sistema deve contenere i seguenti elementi:

- Il AWS CLI, che viene utilizzato per avviare una sessione del portale di AWS accesso prima di eseguire l'applicazione.
- Il AWS config file condiviso che contiene un [\[default\]profilo](#) con un set di valori di configurazione a cui è possibile fare riferimento negli Strumenti per PowerShell. Per trovare la posizione di questo file, consulta l'argomento relativo alla [posizione dei file condivisi](#) nella Guida di riferimento per SDK e strumenti AWS . Tools for PowerShell utilizza il provider di token SSO del profilo per acquisire le credenziali prima di inviare richieste a. AWS Il `sso_role_name` valore, che è un ruolo IAM connesso a un set di autorizzazioni IAM Identity Center, dovrebbe consentire l'accesso ai dati Servizi AWS utilizzati nell'applicazione.

Il seguente config file di esempio mostra un `[default]` profilo configurato con un provider di token SSO. L'impostazione `sso_session` del profilo si riferisce alla sezione `sso-session` denominata. La `sso-session` sezione contiene le impostazioni per avviare una sessione del portale di AWS accesso.

```
[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

La PowerShell sessione deve avere i seguenti moduli installati e importati in modo che la risoluzione SSO possa funzionare:

- `AWS.Tools.SSO`
- `AWS.Tools.SSOIDC`

Se utilizzi una versione precedente di Tools for PowerShell e non disponi di questi moduli, riceverai un errore simile al seguente: «Assembly AWSSDK .SSOIDC could not be found...».

Avvia una sessione del portale di accesso AWS

Prima di eseguire i comandi di accesso Servizi AWS, è necessaria una sessione attiva del portale di AWS accesso in modo che Tools for Windows PowerShell possa utilizzare l'autenticazione IAM Identity Center per risolvere le credenziali. A seconda della durata della sessione configurata, l'accesso alla fine scadrà e gli Strumenti per Windows PowerShell riscontreranno un errore di autenticazione. Per accedere al portale di AWS accesso, esegui il seguente comando in AWS CLI

```
aws sso login
```

Poiché si utilizza il [default] profilo, non è necessario richiamare il comando con l'--profileopzione. Se la configurazione del provider di token SSO utilizza un profilo denominato, il comando è `aws sso login --profile named-profile` invece. Per ulteriori informazioni sui profili denominati, consulta la sezione [Profili](#) nella Guida di riferimento agli AWS SDK e agli strumenti.

Per verificare se hai già una sessione attiva, esegui il AWS CLI comando seguente (con la stessa considerazione per named profile):

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

La risposta a questo comando dovrebbe restituire l'account IAM Identity Center e il set di autorizzazioni configurati nel file config condiviso.

Note

Se hai già una sessione attiva del portale di AWS accesso ed esegui `aws sso login`, non ti verrà richiesto di fornire credenziali.

La procedura di accesso potrebbe richiedere all'utente di consentire l'AWS CLI accesso ai dati. Poiché AWS CLI è basato sull'SDK per Python, i messaggi di autorizzazione possono contenere variazioni del botocore nome.

Esempio

Di seguito è riportato un esempio di come utilizzare IAM Identity Center con gli strumenti per PowerShell. Si presume quanto segue:

- Hai abilitato IAM Identity Center e lo hai configurato come descritto in questo argomento. Le proprietà SSO si trovano nel profilo [default].
- Quando accedi tramite AWS CLI by using `aws sso login`, quell'utente dispone almeno delle autorizzazioni di sola lettura per Amazon S3.
- Alcuni bucket S3 sono disponibili per la visualizzazione da parte dell'utente.

Usa i seguenti PowerShell comandi per visualizzare un elenco dei bucket S3:

```
Install-Module AWS.Tools.Installer
Install-AWSToolsModule S3
# And if using an older version of the AWS Tools for PowerShell:
Install-AWSToolsModule SSO, SS00IDC

# 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.SS00IDC

# 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
```

Come accennato in precedenza, poiché si utilizza il [default] profilo, non è necessario chiamare il `Get-S3Bucket` cmdlet con l'opzione `-ProfileName`. Se la configurazione del provider di token SSO utilizza un profilo denominato, il comando è `Get-S3Bucket -ProfileName named-profile`. Per ulteriori informazioni sui profili denominati, vedere la sezione [Profili](#) nella Guida di riferimento agli AWS SDK e agli strumenti.

Informazioni aggiuntive

- Per ulteriori opzioni sull'autenticazione per gli Strumenti per PowerShell, come l'uso di profili e variabili di ambiente, consulta il capitolo sulla [configurazione](#) nella Guida di riferimento agli AWS SDK e agli strumenti.
- Alcuni comandi richiedono la specificazione di una AWS regione. Esistono diversi modi per farlo, tra cui l'opzione del `-Region` cmdlet, il `[default]` profilo e la variabile di ambiente `AWS_REGION`. Per ulteriori informazioni, consulta [Specificare AWS le regioni](#) questa guida e [l'AWS area geografica nella](#) Guida di riferimento agli AWS SDK e agli strumenti.
- Per ulteriori informazioni sulle best practice, consulta [Best practice per la sicurezza in IAM](#) nella Guida per l'utente di IAM.
- Per creare AWS credenziali a breve termine, consulta [Temporary Security Credentials](#) nella IAM User Guide.
- Per ulteriori informazioni su altri provider di credenziali, consulta l'argomento relativo ai [provider di credenziali standard](#) nella Guida di riferimento per SDK e strumenti AWS .

Specificare AWS le regioni

Esistono due modi per specificare la AWS regione da utilizzare durante l'esecuzione dei AWS Tools for PowerShell comandi:

- Utilizza il parametro `-Region` comune sui singoli comandi.
- Utilizza il comando `Set-DefaultAWSRegion` per impostare una regione predefinita per tutti i comandi.

Molti AWS cmdlet hanno esito negativo se gli Strumenti per Windows non PowerShell riescono a capire quale regione utilizzare. Le eccezioni includono i cmdlet per Amazon S3, [Amazon](#) SES AWS Identity and Access Management e, che utilizzano automaticamente per impostazione predefinita un endpoint globale.

Per specificare la regione per un singolo comando AWS


Aggiungi il parametro `-Region` al comando, come il seguente.

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


Per impostare una regione predefinita per tutti i comandi AWS CLI nella sessione corrente

Dal PowerShell prompt dei comandi, digitare il comando seguente.

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

 Note

Questa impostazione persiste solo nella sessione corrente. Per applicare l'impostazione a tutte le PowerShell sessioni, aggiungi questo comando al tuo PowerShell profilo come hai fatto per il `Import-Module` comando.

Per visualizzare l'area predefinita corrente per tutti i comandi AWS CLI

Dal PowerShell prompt dei comandi, digitare il seguente comando.

```
PS > Get-DefaultAWSRegion
```

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

Per cancellare la regione predefinita corrente per tutti i comandi AWS CLI

Dal PowerShell prompt dei comandi, digitate il seguente comando.

```
PS > Clear-DefaultAWSRegion
```

Per visualizzare un elenco di tutte le regioni disponibili AWS

Dal PowerShell prompt dei comandi, digitare il comando seguente. La terza colonna nell'output di esempio identifica la Regione predefinita per la sessione corrente.

```
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

Alcune Regioni potrebbero essere supportate ma non incluse negli output del cmdlet `Get-AWSRegion`. Ad esempio, questo è talvolta vero per le Regioni che non sono ancora globali. Se non sei in grado di specificare una Regione aggiungendo il parametro `-Region` a un comando, prova invece a specificare la Regione in un endpoint personalizzato, come illustrato nella sezione successiva.

Specificare un endpoint personalizzato o non standard

Specificate un endpoint personalizzato come URL aggiungendo il parametro `-EndpointUrl` common al PowerShell comando Tools for Windows, nel seguente formato di esempio.

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

L'esempio seguente utilizza il cmdlet `Get-EC2Instance`. L'endpoint personalizzato si trova in `us-west-2` o nella Regione Stati Uniti occidentali (Oregon) in questo esempio, ma puoi utilizzare qualsiasi altra Regione AWS supportata, incluse le Regioni che non sono enumerate da `Get-AWSRegion`.

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

Informazioni aggiuntive

Per ulteriori informazioni sulle AWS regioni, consulta la sezione [AWS Regione nella Guida di riferimento agli AWS SDK e agli strumenti](#).

Configurazione dell'identità federata con AWS Tools for PowerShell

Per consentire agli utenti nella tua organizzazione di accedere alle risorse AWS, devi configurare un metodo di autenticazione standard e ripetibile a scopo di sicurezza, di verifica e di conformità con la

capacità di supportare il ruolo e la separazione dell'account. Sebbene sia comune fornire agli utenti l'accesso alle API AWS, senza l'accesso federato alle API sarebbe necessario creare anche utenti AWS Identity and Access Management (IAM), rendendo inefficace l'utilizzo della federazione. Questo argomento descrive il supporto SAML (Security Assertion Markup Language) negli AWS Tools for PowerShell che semplifica la soluzione di accesso federato.

Il supporto per SAML negli AWS Tools for PowerShell consente di offrire agli utenti federati l'accesso ai servizi AWS. SAML è un formato standard aperto basato su XML per la trasmissione di dati di autenticazione e autorizzazione dell'utente tra servizi; in particolare, tra un provider di identità (ad esempio [Active Directory Federation Services](#)) e un fornitore di servizi (come AWS). Per ulteriori informazioni su SAML e su come funziona, consultare la pagina [SAML](#) su Wikipedia o [SAML Technical Specifications](#) sul sito del consorzio della Organization for the Advancement of Structured Information Standards (OASIS). Il supporto SAML negli AWS Tools for PowerShell è compatibile con SAML 2.0.

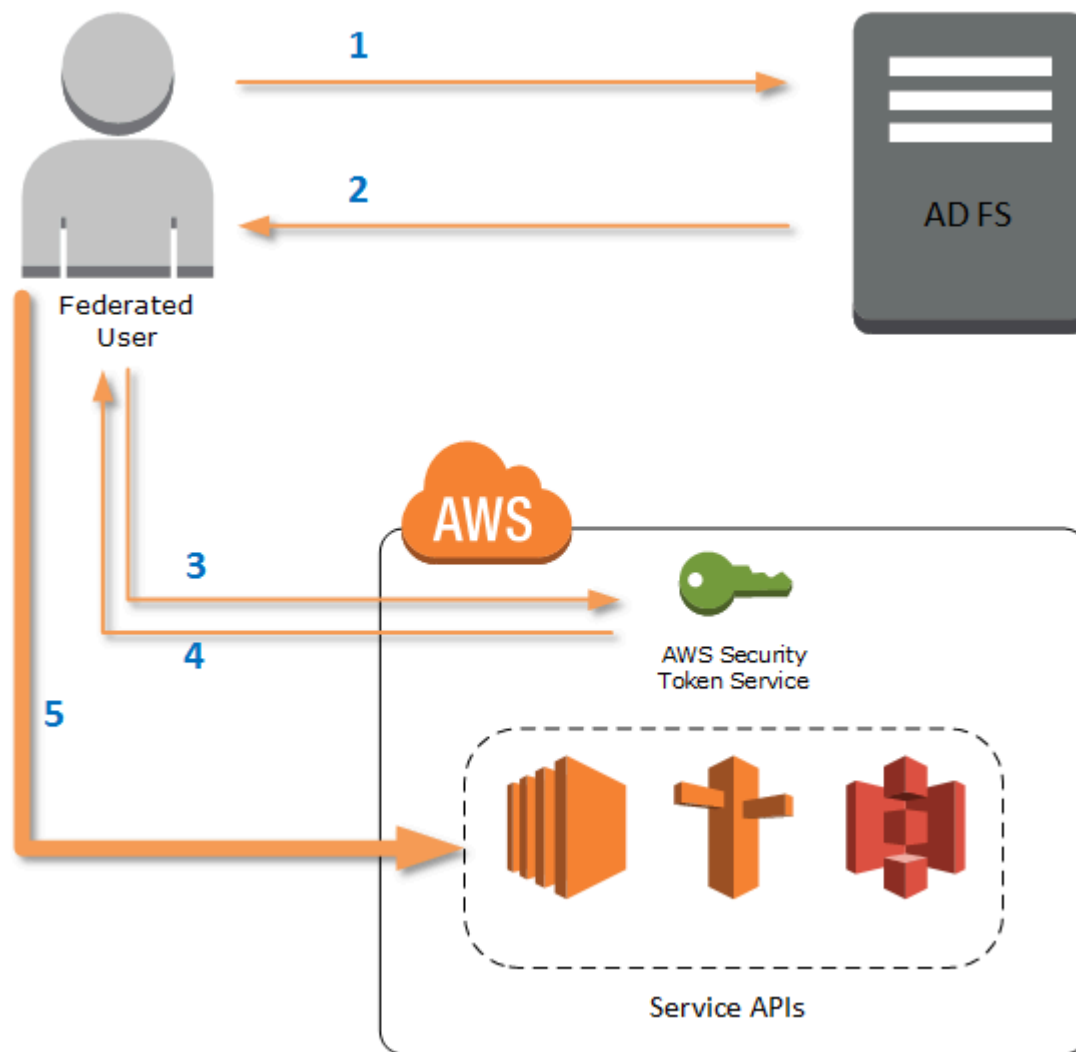
Prerequisiti

È necessario soddisfare i seguenti prerequisiti prima di provare a utilizzare il supporto a SAML per la prima volta.

- Una soluzione di identità federata correttamente integrata con l'account AWS per l'accesso alla console utilizzando solo le credenziali della propria organizzazione. Per ulteriori informazioni su come eseguire questa operazione in modo specifico per Active Directory Federation Services, consulta le [Informazioni sulla federazione basata su SAML 2.0](#) nella Guida per l'utente IAM e il post del blog sull'[abilitazione della federazione ad AWS utilizzando Windows Active Directory, AD FS e SAML 2.0](#). Anche se il blog post copre AD FS 2.0, le fasi sono simili se si esegue AD FS 3.0.
- Versione 3.1.31.0 o più recenti degli AWS Tools for PowerShell installata sulla workstation locale.

In che modo un utente con identità federata ottiene l'accesso federato alle API del servizio AWS

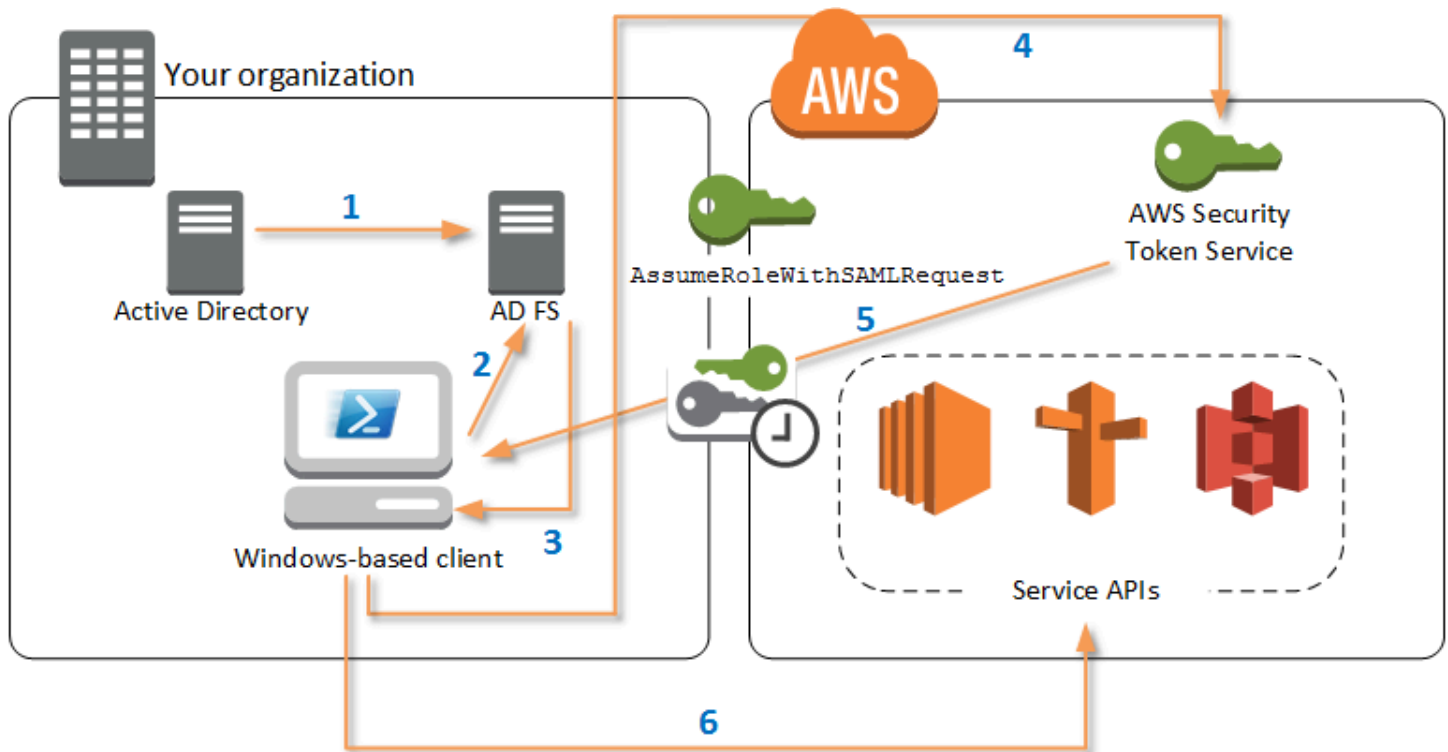
Il processo seguente descrive, a livello generale, come un utente Active Directory (AD) sia federato da AD FS per ottenere l'accesso alle risorse AWS.



1. Il client nel computer dell'utente federato esegue l'autenticazione tramite AD FS.
2. Se l'autenticazione ha esito positivo, AD FS invia all'utente un'asserzione SAML.
3. Il client dell'utente invia l'asserzione SAML al AWS Security Token Service (STS) come parte di una richiesta di federazione SAML.
4. STS restituisce una risposta SAML che contiene le credenziali AWSTemporanee per un ruolo che l'utente può assumere.
5. L'utente accede alle API del servizio AWS includendo tali credenziali temporanee nella richiesta effettuata dagli AWS Tools for PowerShell.

Come funziona il supporto SAML con gli AWS Tools for PowerShell

Questa sezione descrive come i cmdlet degli AWS Tools for PowerShell abilitano la configurazione della federazione delle identità SAML per gli utenti.



1. Gli AWS Tools for PowerShell effettuano l'autenticazione con AD FS utilizzando le credenziali correnti dell'utente Windows oppure in modo interattivo quando l'utente prova a eseguire un cmdlet che richiede credenziali per chiamare AWS.
2. AD FS autentica l'utente.
3. AD FS genera una risposta di autenticazione SAML 2.0 che include un'asserzione; lo scopo dell'asserzione è quello di identificare e fornire informazioni sull'utente. Gli AWS Tools for PowerShell estraggono l'elenco dei ruoli dell'utente autorizzato dall'asserzione SAML.
4. Gli `AssumeRoleWithSAMLRequest` inoltrano la richiesta SAML, tra cui il ruolo richiesto ARN (Amazon Resource Name), a STS tramite la chiamata API AWS Tools for PowerShell.
5. Se la richiesta SAML è valida, STS restituirà una risposta che contiene i parametri `AccessKeyId`, `SecretAccessKey` e `SessionToken` di AWS. Queste credenziali dureranno per 3.600 secondi (1 ora).
6. L'utente dispone ora di credenziali valide per utilizzare qualsiasi API del servizio AWS a cui il ruolo dell'utente è autorizzato ad accedere. Gli AWS Tools for PowerShell applicano automaticamente

queste credenziali per tutte le successive chiamate API AWS e le rinnovano automaticamente alla scadenza.

Note

Quando le credenziali scadono e sono necessarie nuove credenziali, gli AWS Tools for PowerShell eseguono automaticamente una nuova autenticazione con AD FS e ottengono nuove credenziali per l'ora successiva. Per gli account degli utenti aggiunti al dominio, questo processo viene eseguito automaticamente. Per gli account che non sono aggiunti al dominio, gli AWS Tools for PowerShell richiedono agli utenti di inserire le proprie credenziali prima che possano effettuare nuovamente l'autenticazione.

Come utilizzare la configurazione dei cmdlet SAML PowerShell

Gli AWS Tools for PowerShell includono due nuovi cmdlet che forniscono supporto SAML.

- `Set-AWSSamlEndpoint` configura gli endpoint AD FS, assegna un nome descrittivo per l'endpoint e, facoltativamente, descrive il tipo di autenticazione dell'endpoint.
- `Set-AWSSamlRoleProfile` crea o modifica un profilo di account utente che si desidera associare a un endpoint AD FS, identificato specificando il nome descrittivo fornito al cmdlet `Set-AWSSamlEndpoint`. Ogni ruolo del profilo viene mappato a un singolo ruolo che un utente è autorizzato a eseguire.

Come nel caso dei profili delle credenziali AWS, si assegna un nome descrittivo al profilo del ruolo. È possibile utilizzare lo stesso nome descrittivo con il cmdlet `Set-AWSCredential` oppure come valore del parametro `-ProfileName` per qualsiasi cmdlet che invoca le API del servizio AWS.

Apri una nuova sessione degli AWS Tools for PowerShell. Se è in esecuzione PowerShell 3.0 o versioni più recenti, il modulo degli AWS Tools for PowerShell è importato automaticamente ogni volta che esegui uno dei suoi cmdlet. Se è in esecuzione PowerShell 2.0, devi importare manualmente il modulo eseguendo il cmdlet `Import-Module`, come illustrato nell'esempio seguente.

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

Come eseguire i cmdlet **Set-AWSSamlEndpoint** e **Set-AWSSamlRoleProfile**

1. In primo luogo, configurare le impostazioni degli endpoint per il sistema AD FS. Il modo più semplice per eseguire questa operazione è quello di archiviare l'endpoint in una variabile, come illustrato in questa fase. Assicurarsi di sostituire il segnaposto dell'ID dell'account e il nome dell'host dell'AD FS con il proprio ID dell'account e il nome dell'host dell'AD FS. Specificare il nome dell'host dell'AD FS nel parametro Endpoint.

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

2. Per creare le impostazioni dell'endpoint, eseguire il cmdlet **Set-AWSSamlEndpoint** specificando il valore corretto per il parametro **AuthenticationType**. I valori validi includono Basic, Digest, Kerberos, Negotiate e NTLM. Se non viene specificato questo parametro, il valore predefinito è Kerberos.

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

Il cmdlet restituisce il nome descrittivo assegnato utilizzando il parametro **-StoreAs**, in modo che sia possibile utilizzarlo quando si esegue **Set-AWSSamlRoleProfile** nella riga successiva.

3. Ora, esegui il cmdlet **Set-AWSSamlRoleProfile** per effettuare l'autenticazione con il fornitore di identità AD FS e ottenere il set dei ruoli (nell'asserzione SAML) che l'utente è autorizzato a eseguire.

Il cmdlet **Set-AWSSamlRoleProfile** utilizza il set di ruoli restituito per richiedere all'utente di selezionare un ruolo per associarlo con il profilo specificato o convalidare i dati forniti del ruolo in parametri se presenti (in caso contrario all'utente viene richiesto di scegliere). Se l'utente è autorizzato per un solo ruolo, il cmdlet associa il ruolo con il profilo automaticamente, senza richiedere l'intervento dell'utente. Non è necessario fornire credenziali per configurare un profilo per l'utilizzo aggiunto al dominio.

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

In alternativa, per gli account non appartenenti a un dominio, è possibile fornire le credenziali Active Directory e quindi selezionare un ruolo AWS a cui l'utente ha accesso, come illustrato nella riga seguente. Questa funzione è utile se si dispone di diversi account utente Active Directory

per distinguere i ruoli all'interno della propria organizzazione (ad esempio per le funzioni di amministrazione).

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

4. In entrambi i casi, il cmdlet `Set-AWSSamlRoleProfile` richiede di scegliere quale ruolo deve essere memorizzato nel profilo. Il seguente esempio mostra due ruoli disponibili: `ADFS-Dev` e `ADFS-Production`. I ruoli IAM sono associati alle tue credenziali di accesso AD dall'amministratore di AD FS.

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

In alternativa, puoi specificare un ruolo senza il prompt, inserendo i parametri `RoleARN`, `PrincipalARN` e `NetworkCredential` opzionale. Se il ruolo specificato non è elencato nell'asserzione restituita dall'autenticazione, all'utente viene richiesto di scegliere tra i ruoli disponibili.

```
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. È possibile creare profili per tutti i ruoli in un unico comando aggiungendo il parametro `StoreAllRoles`, come mostrato nel codice seguente. Si noti che il nome del ruolo viene usato come nome del profilo.

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


Come utilizzare i profili del ruolo per eseguire cmdlet che richiedono credenziali AWS

Per eseguire i cmdlet che richiedono le credenziali AWS, puoi utilizzare i profili dei ruoli definiti nel file delle credenziali AWS condiviso. Fornisci il nome di un profilo del ruolo a `Set-AWSCredential` (o come valore per qualsiasi parametro `ProfileName` negli AWS Tools for PowerShell) in modo da ottenere automaticamente credenziali AWS temporanee per il ruolo descritto nel profilo.

Anche se si utilizza un solo profilo del ruolo alla volta, è possibile passare tra i profili all'interno di una sessione shell. Il cmdlet `Set-AWSCredential` non autentica e non ottiene le credenziali quando viene eseguito di per sé; il cmdlet registra il desiderio di utilizzare un specifico profilo di ruolo. Finché esegui un cmdlet che richiede le credenziali AWS, non occorre l'autenticazione o la richiesta di credenziali.

Ora puoi utilizzare le credenziali AWS temporanee ottenute con il profilo `SAMLDemoProfile` per utilizzare le API del servizio AWS. Le seguenti sezioni mostrano degli esempi su come utilizzare i profili di ruolo.

Esempio 1: Impostare un ruolo predefinito con `Set-AWSCredential`

Questo esempio imposta un ruolo predefinito per una sessione degli AWS Tools for PowerShell utilizzando `Set-AWSCredential`. Quindi, è possibile eseguire i cmdlet che richiedono le credenziali e sono autorizzati dal ruolo specificato. In questo esempio vengono elencate tutte le istanze Amazon Elastic Compute Cloud nella Regione Stati Uniti occidentali (Oregon) associate al profilo specificato con il cmdlet `Set-AWSCredential`.

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

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

Esempio 2: Cambiare il profilo del ruolo durante una sessione PowerShell

In questo esempio vengono elencati tutti i bucket Amazon S3 disponibili nell'account AWS del ruolo associato al profilo `SAMLDemoProfile`. L'esempio illustra che anche se è possibile che sia stato precedentemente utilizzato un altro profilo nella sessione degli AWS Tools for PowerShell, puoi modificare i profili specificando un valore diverso per il parametro `-ProfileName` con i cmdlet che lo supportano. Questo è un processo comune per gli amministratori che gestiscono Amazon S3 dalla riga di comando di PowerShell.

```
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
```

Si noti che il cmdlet `Get-S3Bucket` specifica il nome del profilo creato eseguendo il cmdlet `Set-AWSSamlRoleProfile`. Questo comando può essere utile se è stato impostato un profilo all'inizio della sessione (per esempio, eseguendo il cmdlet `Set-AWSCredential`) e si voleva utilizzare un profilo di ruolo differente per il cmdlet `Get-S3Bucket`. Il gestore del profilo rende disponibili le credenziali provvisorie al cmdlet `Get-S3Bucket`.

Anche se le credenziali scadono dopo 1 ora (un limite imposto da STS), gli AWS Tools for PowerShell aggiornano automaticamente le credenziali richiedendo una nuova asserzione SAML quando gli strumenti rilevano che le attuali credenziali sono scadute.

Per gli utenti aggiunti al dominio, questo processo viene eseguito senza interruzioni, perché l'attuale identità dell'utente Windows viene utilizzata durante l'autenticazione. Per gli account degli utenti non aggiunti al dominio, gli AWS Tools for PowerShell mostrano una richiesta di credenziali PowerShell richiedendo la password dell'utente. L'utente fornisce le credenziali che sono state utilizzate per la sua nuova autenticazione e per ottenere una nuova asserzione.

Esempio 3: Ottenere istanze in una regione

L'esempio seguente elenca tutte le istanze Amazon EC2 nella Regione Asia Pacifico (Sydney) associate all'account utilizzato dal profilo `ADFS-Production`. Si tratta di un comando utile per restituire tutte le istanze Amazon EC2 in una Regione.

```
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

Ulteriori letture

Per informazioni generali su come implementare l'accesso federato alle API, consultare la pagina [Come implementare una soluzione generale per l'accesso federato API/CLI tramite SAML 2.0](#).

Per domande di assistenza o commenti, visita i forum degli sviluppatori AWS per gli [script PowerShell](#) o lo [sviluppo .NET](#).

Individuazione di cmdlet e alias

Questa sezione illustra come visualizzare l'elenco dei servizi supportati dagli AWS Tools for PowerShell, come mostrare i set di cmdlet forniti dagli AWS Tools for PowerShell a sostegno di questi servizi e come individuare i nomi dei cmdlet alternativi (detti anche alias) per l'accesso a questi servizi.

Individuare i cmdlet

Tutte le operazioni dei servizi AWS (o API) sono documentate nella Guida di riferimento dell'API per ogni servizio. Ad esempio, consulta la [Guida di riferimento dell'API IAM](#). Nella maggior parte dei casi esiste una corrispondenza uno-a-uno tra un'API del servizio AWS e un cmdlet PowerShell AWS. Per ottenere il nome del cmdlet che corrisponde al nome di un'API del servizio AWS, esegui il cmdlet AWS `Get-AWSCmdletName` con il parametro `-ApiOperation` e il nome dell'API del servizio AWS. Ad esempio, per ottenere tutti i possibili nomi di cmdlet che si basano su qualsiasi API `DescribeInstances` del servizio AWS, esegui il comando seguente:

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

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

Il parametro `-ApiOperation` è il parametro predefinito, quindi puoi omettere il nome del parametro. L'esempio seguente è uguale al precedente:

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

Se conosci i nomi sia dell'API che del servizio, puoi includere il parametro `-Service` con il prefisso del nome del cmdlet o con parte del nome del servizio AWS. Ad esempio, il prefisso del nome del cmdlet per Amazon EC2 è EC2. Per ottenere il nome del cmdlet che corrisponde all'API `DescribeInstances` nel servizio Amazon EC2, esegui uno dei seguenti comandi. Sono tutti risultati nello stesso output:

```
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

I valori del parametro in questi comandi fanno distinzione tra maiuscola e minuscola.

Se non conosci il nome dell'API del servizio AWS desiderato o il nome del servizio AWS, puoi utilizzare il parametro `-ApiOperation` con il modello da abbinare e il parametro `-MatchWithRegex`. Ad esempio, per ottenere tutti i nomi dei cmdlet disponibili che contengono `SecurityGroup`, esegui il comando seguente:

```
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	

Se conosci il nome del servizio AWS ma non il nome dell'API del servizio AWS, includi sia il parametro `-MatchWithRegex` che il parametro `-Service` per limitare la ricerca a un singolo servizio. Ad esempio, per ottenere tutti i nomi dei cmdlet che contengono SecurityGroup solo nel servizio Amazon EC2, esegui il comando seguente

```
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 Amazon Elastic Compute Cloud EC2	DeleteSecurityGroup
Revoke-EC2SecurityGroupEgress Amazon Elastic Compute Cloud EC2	RevokeSecurityGroupEgress
Revoke-EC2SecurityGroupIngress Amazon Elastic Compute Cloud EC2	RevokeSecurityGroupIngress
Update-EC2SecurityGroupRuleEgressDescription Amazon Elastic Compute Cloud EC2	UpdateSecurityGroupRuleDescriptionsEgress
Update-EC2SecurityGroupRuleIngressDescription UpdateSecurityGroupRuleDescriptionsIngress	Amazon Elastic Compute Cloud EC2

Se conosci il nome del comando AWS Command Line Interface (AWS CLI), puoi usare il parametro `-AwsCliCommand` e il nome del comando AWS CLI desiderato per ottenere il nome del cmdlet basato sulla stessa API. Ad esempio, per ottenere il nome del cmdlet che corrisponde alla chiamata `authorize-security-group-ingress` del comando AWS CLI nel servizio Amazon EC2, esegui il comando seguente:

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

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

Il cmdlet `Get-AWSCmdletName` necessita solo di una parte sufficiente del nome del comando AWS CLI per identificare il servizio e l'API AWS.

Per ottenere un elenco di tutti i cmdlet nei Tools for PowerShell Core, esegui il cmdlet `Get-Command` di PowerShell, come mostrato nell'esempio seguente.

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

Puoi eseguire lo stesso comando con `-Module AWSPowerShell` per visualizzare i cmdlet negli AWS Tools for Windows PowerShell.

Il cmdlet `Get-Command` genera l'elenco di cmdlet in ordine alfabetico. Per impostazione predefinita, l'elenco è ordinato in base al verbo PowerShell anziché al nome PowerShell.

Per ordinare i risultati in base al servizio, esegui il comando seguente.

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

Per filtrare i cmdlet restituiti dal cmdlet `Get-Command`, indirizza l'output al cmdlet `Select-String` di PowerShell. Ad esempio, per visualizzare il set dei cmdlet da utilizzare con le Regioni AWS, esegui il comando seguente:

```
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
```

È inoltre possibile trovare i cmdlet per un servizio specifico filtrando per il prefisso del servizio dei sostantivi del cmdlet. Per visualizzare l'elenco dei prefissi del servizio disponibili, esegui `Get-AWSPowerShellVersion -ListServiceVersionInfo`. L'esempio seguente illustra i cmdlet che supportano il servizio Amazon CloudWatch Events.

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

CommandType	Name	Version	Source
-----	----	-----	-----
Cmdlet	Add-CWEResourceTag AWSPowerShell.NetCore	3.3.563.1	
Cmdlet	Disable-CWEEventSource AWSPowerShell.NetCore	3.3.563.1	
Cmdlet	Disable-CWERule AWSPowerShell.NetCore	3.3.563.1	
Cmdlet	Enable-CWEEventSource AWSPowerShell.NetCore	3.3.563.1	
Cmdlet	Enable-CWERule AWSPowerShell.NetCore	3.3.563.1	
Cmdlet	Get-CWEEventBus AWSPowerShell.NetCore	3.3.563.1	
Cmdlet	Get-CWEEventBusList AWSPowerShell.NetCore	3.3.563.1	

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-CWRule	3.3.563.1
	AWSPowerShell.NetCore	
Cmdlet	Write-CWETarget	3.3.563.1
	AWSPowerShell.NetCore	

Nomi dei cmdlet e degli alias

I cmdlet negli AWS Tools for PowerShell per ciascun servizio sono basati sui metodi forniti dall'SDK AWS per il servizio. Tuttavia, a causa delle convenzioni di denominazione obbligatorie di PowerShell, il nome di un cmdlet può risultare diverso rispetto al nome del metodo o della chiamata API su cui si basa. Ad esempio, il cmdlet `Get-EC2Instance` si basa sul metodo `Amazon EC2DescribeInstances`.

In alcuni casi, il nome del cmdlet può essere simile a un nome del metodo, ma potrebbe effettivamente eseguire una funzione differente. Ad esempio, il metodo `Amazon S3GetObject` recupera un oggetto Amazon S3. Tuttavia, il cmdlet `Get-S3Object` restituisce le informazioni su un oggetto Amazon S3 anziché l'oggetto stesso.

```
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
```

Per ottenere un oggetto S3 con gli AWS Tools for PowerShell, esegui il cmdlet `Read-S3Object`.

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

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

Note

La Guida del cmdlet per un cmdlet AWS fornisce il nome dell'API dell'SDK AWS su cui è basato il cmdlet.

Per ulteriori informazioni sui verbi standard di PowerShell e sui loro significati, consulta [Verbi approvati per comandi PowerShell](#).

Tutti i cmdlet AWS che utilizzano il verbo Remove e il cmdlet Stop-EC2Instance quando si aggiunge il parametro -Terminate richiedono una conferma prima di continuare. Per ignorare la conferma, aggiungere il parametro -Force per il comando.

Important

I cmdlet AWS non supportano lo switch -WhatIf.

Alias

La configurazione degli AWS Tools for PowerShell installa un file alias che contiene gli alias per molti cmdlet AWS. Questi alias potrebbero risultare più intuitivi dei nomi dei cmdlet. Ad esempio, i nomi dei servizi e i nomi dei metodi dell'SDK AWS sostituiscono i verbi e i nomi di PowerShell in alcuni alias. Un esempio è l'alias EC2-DescribeInstances.

Altri alias utilizzano verbi che, sebbene non seguano le convenzioni standard della PowerShell, potrebbero essere più descrittivi dell'operazione effettiva. Ad esempio, il file alias associa l'alias Get-S3Content al cmdlet Read-S3Object.

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

Il file alias si trova nella directory di installazione di AWS Tools for PowerShell. Per caricare gli alias nell'ambiente, eseguire il file dot-source. Il seguente è un esempio basato su Windows.

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

Per una shell Linux o macOS, potrebbe essere simile a questo:

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

Per visualizzare tutti gli alias AWS Tools for PowerShell, esegui il comando seguente. Questo comando utilizza l'alias `?` per il cmdlet `Where-Object` di PowerShell e la proprietà `Source` per filtrare solo gli alias provenienti dal modulo `AWSPowerShell.NetCore`.

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

CommandType	Name	Version	Source
-----	----	-----	-----
Alias	Add-ASInstances	3.3.343.0	AWSPowerShell
Alias	Add-CTTag	3.3.343.0	AWSPowerShell
Alias	Add-DPTags	3.3.343.0	AWSPowerShell
Alias	Add-DSIpRoutes	3.3.343.0	AWSPowerShell
Alias	Add-ELBTags	3.3.343.0	AWSPowerShell
Alias	Add-EMRTag	3.3.343.0	AWSPowerShell
Alias	Add-ESTag	3.3.343.0	AWSPowerShell
Alias	Add-MLTag	3.3.343.0	AWSPowerShell
Alias	Clear-AWSCredentials	3.3.343.0	AWSPowerShell
Alias	Clear-AWSDefaults	3.3.343.0	AWSPowerShell
Alias	Dismount-ASInstances	3.3.343.0	AWSPowerShell
Alias	Edit-EC2Hosts	3.3.343.0	AWSPowerShell
Alias	Edit-RSClusterIamRoles	3.3.343.0	AWSPowerShell
Alias	Enable-ORGAllFeatures	3.3.343.0	AWSPowerShell
Alias	Find-CTEvents	3.3.343.0	AWSPowerShell
Alias	Get-ASACases	3.3.343.0	AWSPowerShell
Alias	Get-ASAccountLimits	3.3.343.0	AWSPowerShell

Alias AWSPowerShell	Get-ASACommunications	3.3.343.0
Alias AWSPowerShell	Get-ASAServices	3.3.343.0
Alias AWSPowerShell	Get-ASASeverityLevels	3.3.343.0
Alias AWSPowerShell	Get-ASATrustedAdvisorCheckRefreshStatuses	3.3.343.0
Alias AWSPowerShell	Get-ASATrustedAdvisorChecks	3.3.343.0
Alias AWSPowerShell	Get-ASATrustedAdvisorCheckSummaries	3.3.343.0
Alias AWSPowerShell	Get-ASLifecycleHooks	3.3.343.0
Alias AWSPowerShell	Get-ASLifecycleHookTypes	3.3.343.0
Alias AWSPowerShell	Get-AWSCredentials	3.3.343.0
Alias AWSPowerShell	Get-CDApplications	3.3.343.0
Alias AWSPowerShell	Get-CDDeployments	3.3.343.0
Alias AWSPowerShell	Get-CFCloudFrontOriginAccessIdentities	3.3.343.0
Alias AWSPowerShell	Get-CFDistributions	3.3.343.0
Alias AWSPowerShell	Get-CFGConfigRules	3.3.343.0
Alias AWSPowerShell	Get-CFGConfigurationRecorders	3.3.343.0
Alias AWSPowerShell	Get-CFGDeliveryChannels	3.3.343.0
Alias AWSPowerShell	Get-CFInvalidations	3.3.343.0
Alias AWSPowerShell	Get-CFNAccountLimits	3.3.343.0
Alias AWSPowerShell	Get-CFNStackEvents	3.3.343.0
...		

Per aggiungere i propri alias a questo file, potrebbe essere necessario aumentare il valore della [Variabile di preferenza](#) `$MaximumAliasCount` di PowerShell a un valore superiore a 5500. Il valore

predefinito è 4096; è possibile aumentarlo a un massimo di 32768. Per farlo, eseguire il comando seguente.

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

Per verificare che la modifica abbia avuto successo, inserire il nome della variabile per visualizzarne il valore corrente.

```
PS > $MaximumAliasCount  
32768
```

Indirizzamento e \$AWSHistory

Per le chiamate di servizio AWS che restituiscono raccolte, gli oggetti all'interno della raccolta sono enumerati alla pipeline. Gli oggetti risultato che contengono campi aggiuntivi oltre la raccolta e che non sono campi di controllo di paginazione dispongono di questi campi aggiunti come proprietà della nota per le chiamate. Queste proprietà della nota sono registrate nella nuova variabile di sessione `$AWSHistory`, se fosse necessario accedere a questi dati. La variabile `$AWSHistory` è descritta nella sezione successiva.

Note

Nelle versioni dei Tools for Windows PowerShell precedenti alla 1.1, veniva restituito l'oggetto di raccolta, il che richiedeva l'uso di `foreach {$_getenumerator()}` per continuare l'indirizzamento.

Examples (Esempi)

Il seguente esempio restituisce un elenco delle Regioni AWS e delle AMI Amazon EC2 in ogni Regione.

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

Il seguente esempio arresta tutte le istanze Amazon EC2 nella Regione corrente predefinita.

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

Poiché le raccolte sono enumerate alla pipeline, l'output di un determinato cmdlet potrebbe essere `$null`, un singolo oggetto o una raccolta. Se è una raccolta, è possibile utilizzare la proprietà `.Count` per determinare le dimensioni della raccolta stessa. Tuttavia, la proprietà `.Count` non è presente quando viene emesso un singolo oggetto. Se lo script deve determinare, in modo coerente, il numero di oggetti emessi, puoi usare la proprietà `EmittedObjectsCount` dell'ultimo valore di comando in `$AWSHistory`.

\$AWSHistory

Per supportare meglio l'indirizzamento, l'output dai cmdlet AWS non viene rimodellato per includere la risposta del servizio e le istanze del risultato come Proprietà della nota sull'oggetto raccolta restituito. Al contrario, per le chiamate che emettono una singola raccolta come output, la raccolta è ora enumerata alla pipeline della PowerShell. Ciò significa che la risposta dell'SDK AWS e i dati del risultato non possono esistere nella pipe, poiché non esiste un oggetto raccolta a essi collegato che li contenga.

Sebbene non siano necessari alla maggior parte degli utenti, questi dati possono essere utili a scopi diagnostici, perché consentono di vedere esattamente ciò che è stato inviato e ricevuto dalle chiamate effettuate dal cmdlet al servizio AWS sottostante.

A partire dalla versione 1.1, questi e altri dati sono ora disponibili in una nuova variabile shell denominata `$AWSHistory`. Questa variabile mantiene una registrazione delle chiamate del cmdlet AWS e delle risposte del servizio ricevute per ogni chiamata. Facoltativamente, questa cronologia può essere configurata anche per registrare le richieste di servizio effettuate da ogni cmdlet. Inoltre, da ogni voce possono essere ottenuti altri dati utili, ad esempio il tempo di esecuzione complessivo del cmdlet. Per motivi di sicurezza, le richieste e le risposte che contengono dati sensibili non vengono registrate per impostazione predefinita. Tuttavia, la cronologia può essere configurata per sovrascrivere questo comportamento, se necessario. Per ulteriori informazioni, consulta il cmdlet `Set-AWSHistoryConfiguration` riportato di seguito.

Ogni voce nell'elenco `$AWSHistory.Commands` è di tipo `AWSCmdletHistory`. Questo tipo ha i seguenti membri utili:

CmdletName

Nome del cmdlet.

CmdletStart

DateTime di esecuzione del cmdlet.

CmdletEnd

DateTime di fine di tutti i processi del cmdlet.

Richieste

Se la richiesta di registrazione è abilitata, vengono elencate le ultime richieste di servizio.

Risposte

Elenco delle ultime risposte di servizio ricevute.

LastServiceResponse

Helper per restituire la più recente risposta del servizio.

LastServiceRequest

Helper per restituire la più recente richiesta del servizio, se disponibile.

Tieni presente che la variabile `$AWSHistory` non viene creata finché non viene utilizzato un cmdlet AWS che effettua una chiamata di servizio. Viene valutata su `$null` fino a quel momento.

Note

Le versioni precedenti dei Tools for Windows PowerShell restituivano dati relativi alle risposte del servizio come proprietà `Note` sull'oggetto restituito. Queste sono ora disponibili sulle voci di risposta registrate per ogni invocazione nell'elenco.

Set-AWSHistoryConfiguration

Una invocazione cmdlet può contenere zero o più richieste di servizio e voci di risposta. Per limitare l'impatto della memoria, l'elenco `$AWSHistory` mantiene solo la registrazione delle ultime cinque esecuzioni di cmdlet come impostazione predefinita e, per ognuna, le ultime cinque risposte di servizio (e, se abilitate, le ultime cinque richieste di servizio). È possibile modificare questi limiti predefiniti eseguendo il cmdlet `Set-AWSHistoryConfiguration`. Questo permette di controllare la dimensione dell'elenco e se le richieste di servizio sono state registrate.

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


Tutti i parametri sono opzionali:

Il parametro `MaxCmdletHistory` imposta il numero massimo di cmdlet che può essere monitorato in qualsiasi momento. Il valore 0 disattiva la registrazione dell'attività del cmdlet AWS. Il parametro `MaxServiceCallHistory` imposta il numero massimo di risposte di servizio (e/o richieste) che vengono registrate per ogni cmdlet. Il parametro `RecordServiceRequests`, se specificato, consente di attivare il monitoraggio delle richieste di servizio per ogni cmdlet. Il parametro `IncludeSensitiveData`, se specificato, attiva il tracciamento delle risposte e delle richieste di servizio (se tracciate) che contengono dati sensibili per ogni cmdlet.

Se si esegue senza parametri, `Set-AWSHistoryConfiguration` disattiva semplicemente , qualsiasi precedente registrazione della richiesta, lasciando invariata la dimensione corrente dell'elenco.

Per cancellare tutte le voci nell'elenco della cronologia corrente, eseguire il cmdlet `Clear-AWSHistory`.

Esempi di `$AWSHistory`

Enumerare nella pipeline i dettagli dei cmdlet AWS contenuti nell'elenco.

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

Accedere ai dettagli dell'ultimo cmdlet AWS eseguito:

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

Accedere ai dettagli dell'ultima risposta del servizio ricevuta dall'ultimo cmdlet AWS eseguito. Se un cmdlet AWS esegue l'impaginazione dell'output, può effettuare più chiamate di servizio per ottenere tutti i dati o la quantità massima di dati (determinata dai parametri del cmdlet).

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

Accedere ai dettagli dell'ultima richiesta fatta (come prima, un cmdlet può fare più di una richiesta, se sta impaginando per conto dell'utente). Restituisce `$null`, a meno che il tracciamento della richiesta di servizio non sia abilitato.

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

Completamento della pagina automatico per operazioni che restituiscono pagine multiple

Per le API di servizio che impongono un numero totale massimo predefinito di oggetti restituiti per una specifica chiamata o che supportano set di risultati impaginabili, tutti i cmdlet sono impostati su "pagina di completamento" per impostazione predefinita. Ogni cmdlet effettua tutte le chiamate necessarie a tuo nome per restituire l'intero set di dati alla pipeline.

In questo esempio, che utilizza `Get-S3Object`, la variabile `$c` contiene istanze `S3Object` per ogni chiave nel bucket `test`, potenzialmente un set di dati di dimensioni molto grandi.

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

Se desideri mantenere il controllo della quantità di dati restituiti, puoi continuare a utilizzare i parametri sui singoli cmdlet (ad esempio, `MaxKey` su `Get-S3Object`) oppure puoi gestire in modo esplicito l'impaginazione utilizzando una combinazione di parametri di impaginazione nei cmdlet e i dati inseriti nella variabile `$AWSHistory` per ottenere i dati token del servizio successivo. L'esempio seguente utilizza il parametro `MaxKeys` per limitare il numero di istanze `S3Object` restituite a non più delle prime 500 trovate nel bucket.

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

Per sapere se era disponibile una maggiore quantità di dati ma non è stata restituita, utilizzare la voce variabile di sessione `$AWSHistory` che ha registrato le chiamate di servizio effettuate da cmdlet.

Se la seguente espressione restituisce `$true`, puoi trovare il contrassegno `next` per i successivi set di risultati utilizzando `$AWSHistory.LastServiceResponse.NextMarker`:

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

Per controllare manualmente l'impaginazione con `Get-S3Object`, utilizzare una combinazione di parametri `MaxKey` e `Marker` per il cmdlet e le note `IsTruncated/NextMarker` sull'ultima risposta di registrazione. Nel seguente esempio, la variabile `$c` contiene fino a un massimo di 500 istanze `S3Object` per i successivi 500 oggetti trovati nel bucket dopo l'avvio del contrassegno del prefisso della chiave specificato.

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

Risoluzione di credenziali e profili

Ordine di ricerca credenziali

Quando esegui un comando, AWS Tools for PowerShell cerca le credenziali nell'ordine seguente. Si interrompe quando trova le credenziali utilizzabili.

1. Credenziali letterali incorporate come parametri nella riga di comando.

Consigliamo di utilizzare i profili anziché inserire credenziali letterali nelle righe di comando.

2. Un nome specifico del profilo o la posizione del profilo.
 - Se specifichi solo un nome del profilo, il comando cerca il profilo specificato nell'archivio SDK AWS e, se non esiste, cerca il profilo specificato nel file delle credenziali AWS condivise nella posizione predefinita.
 - Se specifichi solo una posizione del profilo, il comando cerca il profilo default dal file delle credenziali.
 - Se specifichi sia un nome sia un percorso, il comando cerca il profilo specificato nel file delle credenziali.

Se il profilo specificato o il percorso non è stato trovato, il comando genera un'eccezione. La ricerca procede nelle seguenti fasi solo se non hai specificato un profilo o una posizione.

3. Credenziali specificate dal parametro `-Credential`.
4. Il profilo di sessione, se esiste.
5. Il profilo predefinito nell'ordine seguente:
 - a. Il profilo di default nell'archivio SDK AWS.
 - b. Il profilo di default nel file delle credenziali AWS condiviso.
 - c. Il profilo di AWS PS Default nell'archivio SDK AWS.
6. Se il comando è in esecuzione su un'istanza Amazon EC2 configurata per l'utilizzo di un ruolo IAM, le credenziali temporanee dell'istanza EC2 a cui si accede dal profilo dell'istanza.

Per ulteriori informazioni sull'utilizzo dei ruoli IAM per le istanze Amazon EC2, consulta [l'AWS SDK for .NET](#).

Se la ricerca ha esito negativo per individuare le credenziali specificate, il comando genera un'eccezione.

Ulteriori informazioni su utenti e ruoli

Per eseguire i comandi degli strumenti per PowerShell su AWS, devi disporre di una combinazione di utenti, set di autorizzazioni e ruoli di servizio appropriati per le attività.

Gli utenti, i set di autorizzazioni e i ruoli di servizio specifici creati e il modo in cui li utilizzi dipenderanno dai tuoi requisiti. Di seguito sono riportate alcune informazioni aggiuntive sul motivo per cui potrebbero essere utilizzati e su come crearli.

Utenti e set di autorizzazioni

Sebbene sia possibile utilizzare un account utente IAM con credenziali a lungo termine per accedere ai servizi AWS, questa non è più una best practice e dovrebbe essere evitata. Anche durante lo sviluppo, è consigliabile creare utenti e set di autorizzazioni in AWS IAM Identity Center e utilizzare credenziali temporanee fornite da un'origine di identità.

Per lo sviluppo, puoi utilizzare l'utente che hai creato o assegnato in [Configurazione dell'autenticazione degli strumenti](#). Se disponi delle autorizzazioni AWS Management Console appropriate, puoi anche creare diversi set di autorizzazioni con il privilegio minimo per quell'utente o creare nuovi utenti specificamente per progetti di sviluppo, fornendo set di autorizzazioni con il privilegio minimo. L'eventuale operazione scelta dipenderà dalle circostanze.


Per ulteriori informazioni su questi utenti e set di autorizzazioni e su come crearli, consulta l'argomento relativo [all'autenticazione e all'accesso](#) nella Guida di riferimento relativa a SDK e strumenti AWS e l'argomento relativo [alle nozioni di base](#) nella Guida per l'utente AWS IAM Identity Center.

Ruoli di servizio

Puoi impostare un ruolo di servizio AWS per accedere ai servizi AWS per conto degli utenti. Questo tipo di accesso è appropriato se più persone eseguiranno l'applicazione in remoto, ad esempio su un'istanza Amazon EC2 creata a tale scopo.

Il processo di creazione di un ruolo di servizio varia a seconda della situazione, ma è essenzialmente simile a quanto indicato di seguito.

1. Accedi alla AWS Management Console e apri la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Selezionare Roles (Ruoli), quindi selezionare Create role (Crea ruolo).
3. Scegli il servizio AWS, quindi come esempio trova e seleziona EC2 e scegli il caso d'uso EC2.
4. Scegli Successivo e seleziona le [policy appropriate](#) per i servizi AWS che verranno utilizzati dall'applicazione.

 Warning

NON scegliere la policy AdministratorAccess perché tale policy applica le autorizzazioni di lettura e scrittura per quasi tutti i contenuti associati al tuo account.

5. Seleziona Successivo. Immetti un valore in Nome ruolo e in Descrizione e quindi immetti qualsiasi altro tag desiderato.


Puoi ulteriori informazioni sui tag, consulta [Controllo dell'accesso alle risorse AWS mediante i tag](#) nella [Guida per l'utente di IAM](#).

6. Scegliere Crea ruolo.

Per ulteriori informazioni di alto livello sui ruoli IAM, consulta [Identità IAM \(utenti, gruppi di utenti e ruoli\)](#) nella [Guida per l'utente di IAM](#). Per informazioni dettagliate sui ruoli, consulta l'argomento [Ruoli IAM](#).

Uso delle credenziali legacy

Negli argomenti di questa sezione vengono fornite informazioni sull'utilizzo di credenziali a lungo termine o a breve termine senza utilizzare AWS IAM Identity Center.

 Warning

Per evitare rischi per la sicurezza, non utilizzare gli utenti IAM per l'autenticazione quando sviluppi software creato ad hoc o lavori con dati reali. Utilizza invece la federazione con un provider di identità come [AWS IAM Identity Center](#).

Note

Le informazioni contenute in questi argomenti si riferiscono a scenari in cui è necessario ottenere e gestire manualmente le credenziali a breve o lungo termine. Per ulteriori informazioni sulle credenziali a breve e lungo termine, consulta la pagina relativa agli [altri modi di autenticazione disponibili](#) nella Guida di riferimento per SDK e strumenti AWS. Per le best practice relative alla sicurezza, usa AWS IAM Identity Center, come descritto in [Configurazione dell'autenticazione degli strumenti](#).

Avvertenze e linee guida importanti per le credenziali

Avvertenze per le credenziali

- NON utilizzare le credenziali root dell'account per accedere alle risorse AWS. Queste credenziali forniscono un accesso illimitato all'account e sono difficili da revocare.
- NON inserire chiavi di accesso letterali o informazioni sulle credenziali nei comandi o negli script. In caso contrario, si corre il rischio di esporre accidentalmente le proprie credenziali.
- Si noti che qualsiasi credenziale archiviata nel file `AWS credentials` condiviso viene archiviata in testo normale.

Linee guida aggiuntive per la gestione sicura delle credenziali

Per una discussione generale su come gestire in modo sicuro le credenziali AWS, consulta [Credenziali di sicurezza di AWS](#) in [Riferimenti generali di AWS](#) e [Best practice per la sicurezza e casi d'uso](#) nella [Guida per l'utente IAM](#). Considera inoltre quanto segue:

- Crea utenti aggiuntivi, come gli utenti in IAM Identity Center, e utilizza le loro credenziali invece di utilizzare le credenziali dell'utente root AWS. Le credenziali per altri utenti possono essere revocate, se necessario, o sono temporanee per natura. Inoltre, puoi applicare una policy a ciascun utente per consentire l'accesso solo a determinate risorse e azioni e quindi adottare autorizzazioni basate sul con privilegio minimo.
- Usa [ruoli IAM](#) per le attività di Amazon Elastic Container Service (Amazon ECS).
- Usa [ruoli IAM](#) per le applicazioni in esecuzione sulle istanze Amazon EC2.

Argomenti

- [Uso delle credenziali AWS](#)
- [Credenziali condivise negli AWS Tools for PowerShell](#)

Uso delle credenziali AWS

Ogni comando AWS Tools for PowerShell deve includere un set di credenziali AWS, che vengono utilizzate per effettuare l'accesso crittografico corrispondente alla richiesta del servizio Web. Puoi specificare le credenziali per comando, per sessione o per tutte le sessioni.

Warning

Per evitare rischi per la sicurezza, non utilizzare gli utenti IAM per l'autenticazione quando sviluppi software creato ad hoc o lavori con dati reali. Utilizza invece la federazione con un provider di identità come [AWS IAM Identity Center](#).

Note

Le informazioni contenute in questo argomento riguardano le circostanze in cui è necessario ottenere e gestire manualmente le credenziali a breve o lungo termine. Per ulteriori informazioni sulle credenziali a breve e lungo termine, consulta l'argomento relativo agli [altri modi di autenticazione disponibili](#) nella Guida di riferimento per SDK e strumenti AWS. Per le best practice relative alla sicurezza, usa AWS IAM Identity Center, come descritto in [Configurazione dell'autenticazione degli strumenti](#).

Come buona pratica, evitare di esporre le proprie credenziali, non utilizzare le credenziali letterali in un comando. Invece, creare un profilo per ogni set di credenziali che si desidera utilizzare e memorizzare il profilo in uno dei due archivi delle credenziali. Specificando il profilo corretto per nome nel comando, gli AWS Tools for PowerShell recuperano le credenziali associate. Per una discussione generale su come gestire in modo sicuro le credenziali AWS, consulta [Best practice per la gestione delle chiavi di accesso AWS](#) nella Riferimenti generali di Amazon Web Services.

Note

Hai bisogno di un account AWS per ottenere le credenziali e utilizzare gli AWS Tools for PowerShell. Per creare un AWS account, consulta la pagina [introduttiva dedicata agli utenti AWS alle prime armi](#) nella Guida di riferimento di AWS Account Management.

Argomenti

- [Posizioni degli archivi per credenziali](#)
- [Gestione dei profili](#)
- [Specifiche delle credenziali](#)
- [Ordine di ricerca credenziali](#)
- [Gestione delle credenziali negli AWS Tools for PowerShell Core](#)

Posizioni degli archivi per credenziali

Gli AWS Tools for PowerShell possono utilizzare entrambi gli archivi delle credenziali:

- L'archivio SDK AWS, che crittografa le credenziali e le archivia nella cartella home. In Windows, questo archivio si trova in: `C:\Users\username\AppData\Local\AWSToolkit\RegisteredAccounts.json`.

L'[AWS SDK for .NET](#) e il [Toolkit for Visual Studio](#) possono anche utilizzare l'archivio SDK AWS.

- Il file delle credenziali condiviso, che si trova anche nella tua cartella home, ma archivia le credenziali come testo semplice.

Per impostazione predefinita, i file delle credenziali vengono archiviati qui:

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

L'SDK AWS e la AWS Command Line Interface possono anche utilizzare i file delle credenziali. Se è in esecuzione uno script al di fuori del contesto utente AWS, accertati che il file delle credenziali venga copiato in un percorso in cui tutti gli account utente (sistema locale e utente) possono accedere alle tue credenziali.

Gestione dei profili

I profili consentono di fare riferimento a diversi set di credenziali con AWS Tools for PowerShell. Puoi utilizzare i cmdlet AWS Tools for PowerShell per gestire i profili nell'archivio SDK AWS. Inoltre, puoi gestire i profili nell'archivio SDK AWS utilizzando il [Toolkit for Visual Studio](#) o a livello di programmazione utilizzando l'[AWS SDK for .NET](#). Per istruzioni su come gestire i profili nel file delle credenziali, consulta le [Best practice per la gestione delle chiavi di accesso AWS](#).

Aggiunta di un nuovo profilo

Per aggiungere un nuovo profilo all'archivio SDK AWS, esegui il comando `Set-AWSCredential`. In questo modo la chiave di accesso e la chiave segreta vengono archiviate nel file delle credenziali predefinito sotto il nome del profilo specificato.

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

- `-AccessKey` – L'ID chiave di accesso.
- `-SecretKey` – La chiave segreta.
- `-StoreAs`– Il nome del profilo, deve essere univoco. Per specificare il profilo predefinito, utilizza il nome `default`.

Aggiornamento di un profilo

L'archivio SDK AWS deve essere gestito manualmente. Se successivamente si modificano le credenziali nel servizio, ad esempio utilizzando la [console IAM](#), eseguendo un comando con le credenziali archiviate in locale e questo ha esito negativo con il seguente messaggio di errore:

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

È possibile aggiornare un profilo ripetendo il comando `Set-AWSCredential` per il profilo e passando a esso le nuove chiavi segrete e di accesso.

Elenco dei profili

Puoi controllare l'elenco corrente dei nomi con il seguente comando. In questo esempio, un utente denominato Shirley ha accesso a tre profili che sono tutti archiviati nel file delle credenziali condiviso (~/.aws/credentials).

```
PS > Get-AWSCredential -ListProfileDetail
```

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

Rimozione di un profilo

Per rimuovere un profilo non più necessario, utilizza il comando seguente.

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

Il parametro `-ProfileName` specifica il profilo da eliminare.

Il comando obsoleto [Clear-AWSCredential](#) è ancora disponibile per la compatibilità con le versioni precedenti, ma `Remove-AWSCredentialProfile` è preferibile.

Specifiche delle credenziali

Esistono vari modi per specificare le credenziali. Il modo migliore è identificare un profilo anziché incorporare credenziali letterali nella riga di comando. AWS Tools for PowerShell individua il profilo utilizzando un ordine di ricerca descritto in [Ordine di ricerca credenziali](#).

In Windows, le credenziali AWS archiviate nell'archivio SDK AWS vengono crittografate con l'identità dell'utente di Windows connesso. Non è possibile decrittografarle utilizzando un altro account o utilizzarle su un dispositivo diverso da quello in cui sono state create originariamente. Per eseguire attività che richiedono le credenziali di un altro utente, ad esempio un account utente utilizzato per eseguire un'attività pianificata, configura un profilo delle credenziali, come descritto nella sezione precedente, che è possibile utilizzare quando accedi al computer come tale utente. Accedi come utente che esegue attività per completare le fasi di configurazione delle credenziali e creare un profilo che funzioni per tale utente. Quindi disconnettiti e accedi di nuovo con le tue credenziali per configurare l'attività pianificata.

Note

Utilizzare il parametro comune `-ProfileName` per specificare un profilo. Questo parametro equivale al parametro `-StoredCredentials` nelle versioni precedenti degli AWS Tools for PowerShell. Per garantire la compatibilità con le versioni precedenti, `-StoredCredentials` è ancora supportato.

Profilo predefinito (consigliato)

Tutti gli strumenti di gestione e gli SDK AWS possono trovare le credenziali automaticamente sul tuo computer locale se le credenziali vengono archiviate in un profilo denominato `default`. Ad esempio, se disponi di un profilo denominato `default` sul computer locale, non devi eseguire il cmdlet `Initialize-AWSDefaultConfiguration` o il cmdlet `Set-AWSCredential`. Gli strumenti utilizzano automaticamente i dati della chiave segreta e di accesso archiviati in quel profilo. Per usare una regione AWS diversa dalla regione predefinita (i risultati di `Get-DefaultAWSRegion`), puoi eseguire `Set-DefaultAWSRegion` e specificare una regione.

Se il profilo non è denominato `default`, ma si desidera utilizzarlo come profilo predefinito per la sessione corrente, eseguire `Set-AWSCredential` per impostare il profilo come predefinito.

Anche se l'esecuzione di `Initialize-AWSDefaultConfiguration` consente di specificare un profilo predefinito per ogni sessione di PowerShell, il cmdlet carica le credenziali dal profilo con nome personalizzato, ma sovrascrive il profilo `default` con il profilo denominato.

È consigliabile non eseguire `Initialize-AWSDefaultConfiguration` a meno che tu non stia eseguendo una sessione PowerShell su un'istanza Amazon EC2 che non è stata avviata con un profilo dell'istanza e desideri impostare in modo manuale il profilo delle credenziali. Tieni presente che il profilo delle credenziali in questo caso non contiene le credenziali. Il profilo delle credenziali risultante dall'esecuzione di `Initialize-AWSDefaultConfiguration` su un'istanza EC2 non archivia direttamente le credenziali, ma punta invece ai metadati dell'istanza (che forniscono credenziali temporanee che ruotano automaticamente). Tuttavia, archivia la regione dell'istanza. Un altro scenario che potrebbe richiedere l'esecuzione di `Initialize-AWSDefaultConfiguration` si verifica se desideri eseguire una chiamata su una regione differente rispetto a quella in cui è in esecuzione l'istanza. L'esecuzione di questo comando sostituisce definitivamente la regione archiviata nei metadati delle istanze.

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

Note

Le credenziali di default sono incluse nell'archivio SDK AWS sotto il nome del profilo `default`. Il comando sovrascrive qualsiasi profilo esistente con lo stesso nome.

Se l'istanza EC2 è stata avviata con un profilo dell'istanza, PowerShell ottiene le informazioni della regione e le credenziali AWS dal profilo dell'istanza. Non devi eseguire `Initialize-AWSDefaultConfiguration`. L'esecuzione del cmdlet `Initialize-AWSDefaultConfiguration` in un'istanza EC2 avviata con un profilo dell'istanza non è necessaria, perché per impostazione predefinita utilizza gli stessi dati del profilo dell'istanza già usati da PowerShell.

Profilo di sessione

Utilizzare `Set-AWSCredential` per specificare un profilo predefinito per una sessione particolare. Questo profilo sostituisce ogni profilo predefinito per la durata della sessione. Questo è consigliato se desideri utilizzare un profilo con nome personalizzato nella sessione anziché il profilo `default` corrente.

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

Note

Nelle versioni dei Tools for Windows PowerShell precedenti alla 1.1, il cmdlet `Set-AWSCredential` non funzionava correttamente e sostituiva il profilo specificato da "MyProfileName". Ti consigliamo di utilizzare una versione più recente dei Tools for Windows PowerShell.

Profilo di comando

Su singoli comandi, puoi aggiungere il parametro `-ProfileName` per specificare un profilo che si applica a un solo comando. Questo profilo sostituisce qualsiasi profilo predefinito o di sessione, come illustrato nell'esempio seguente.

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

Note

Quando specifichi un profilo predefinito o di sessione, puoi anche aggiungere un parametro `-Region` per sostituire una sessione predefinita o la regione della sessione. Per ulteriori informazioni, consulta [Specificare AWS le regioni](#). L'esempio seguente specifica un profilo predefinito e la regione.

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

Per impostazione predefinita, il file delle credenziali AWS condiviso viene archiviato nella cartella home dell'utente (`C:\Users\username\.aws` su Windows o `~/.aws` su Linux). Per specificare un file delle credenziali in un percorso diverso, includi il parametro `-ProfileLocation` e specifica il percorso del file delle credenziali. L'esempio seguente specifica un file delle credenziali non predefinito per un determinato comando.

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

Note

Per l'esecuzione di uno script PowerShell durante un periodo in cui normalmente non sei connesso ad AWS, ad esempio l'esecuzione di uno script PowerShell come processo pianificato al di fuori dei normali orari di lavoro, aggiungi il parametro `-ProfileLocation` quando specifichi il profilo che desideri utilizzare e imposta il valore sul percorso del file che archivia le tue credenziali. Per essere certo che gli script degli AWS Tools for PowerShell siano eseguiti con le credenziali dell'account corretto, devi aggiungere il parametro `-ProfileLocation` ogni volta che lo script viene eseguito in un contesto o in un processo che non utilizza un account AWS. È anche possibile copiare il file delle credenziali in un percorso accessibile al sistema locale o ad altri account che lo script utilizza per eseguire le attività.

Ordine di ricerca credenziali

Quando esegui un comando, AWS Tools for PowerShell cerca le credenziali nell'ordine seguente. Si interrompe quando trova le credenziali utilizzabili.

1. Credenziali letterali incorporate come parametri nella riga di comando.

Consigliamo di utilizzare i profili anziché inserire credenziali letterali nelle righe di comando.

2. Un nome specifico del profilo o la posizione del profilo.
 - Se specifichi solo un nome del profilo, il comando cerca il profilo specificato nell'archivio SDK AWS e, se non esiste, cerca il profilo specificato nel file delle credenziali AWS condivise nella posizione predefinita.
 - Se specifichi solo una posizione del profilo, il comando cerca il profilo default dal file delle credenziali.
 - Se specifichi sia un nome sia un percorso, il comando cerca il profilo specificato nel file delle credenziali.

Se il profilo specificato o il percorso non è stato trovato, il comando genera un'eccezione. La ricerca procede nelle seguenti fasi solo se non hai specificato un profilo o una posizione.

3. Credenziali specificate dal parametro `-Credential`.
4. Il profilo di sessione, se esiste.
5. Il profilo predefinito nell'ordine seguente:
 - a. Il profilo di default nell'archivio SDK AWS.
 - b. Il profilo di default nel file delle credenziali AWS condiviso.
 - c. Il profilo di `AWS PS Default` nell'archivio SDK AWS.
6. Se il comando è in esecuzione su un'istanza Amazon EC2 configurata per l'utilizzo di un ruolo IAM, le credenziali temporanee dell'istanza EC2 a cui si accede dal profilo dell'istanza.

Per ulteriori informazioni sull'utilizzo dei ruoli IAM per le istanze Amazon EC2, consulta [l'AWS SDK for .NET](#).

Se la ricerca ha esito negativo per individuare le credenziali specificate, il comando genera un'eccezione.

Gestione delle credenziali negli AWS Tools for PowerShell Core

I cmdlet negli AWS Tools for PowerShell Core accettano le chiavi di accesso e segrete di AWS o i nomi dei profili delle credenziali quando vengono eseguiti, in modo analogo agli AWS Tools for Windows PowerShell. Quando vengono eseguiti su Windows, entrambi i moduli hanno accesso ai file di archivio delle credenziali AWS SDK for .NET (archiviati per ogni utente in `AppData\Local\AWSToolkit\RegisteredAccounts.json`).

Questo file memorizza le proprie chiavi in formato crittografato e non può essere utilizzato su un altro computer. È il primo file in cui AWS Tools for PowerShell cerca un profilo delle credenziali ed è anche il file in cui gli AWS Tools for PowerShell archiviano i profili delle credenziali. Per ulteriori informazioni sul file dell'archivio delle credenziali AWS SDK for .NET, consulta l'articolo relativo alla [Configurazione delle credenziali AWS](#). Il modulo Tools for Windows PowerShell attualmente non supporta la scrittura di credenziali in altri file o altre posizioni.

Entrambi i moduli sono in grado di leggere i profili dal file delle credenziali AWS condiviso che è utilizzato da altri SDK AWS e dalla AWS CLI. In Windows, la posizione predefinita per questo file è `C:\Users\<userid>\.aws\credentials`. Su piattaforme non Windows, questo file viene memorizzato in `~/.aws/credentials`. Il parametro `-ProfileLocation` può essere utilizzato per puntare a un nome di file non predefinito o a una posizione del file.

L'archivio delle credenziali SDK contiene le credenziali in forma crittografata utilizzando le API di crittografia di Windows. Queste API non sono disponibili per altre piattaforme, perciò il modulo AWS Tools for PowerShell Core utilizza il file delle credenziali AWS condiviso in modo esclusivo e supporta la scrittura di nuovi profili delle credenziali nel file delle credenziali condiviso.

I seguenti script di esempio che utilizzano il cmdlet `Set-AWSCredential` mostrano le opzioni per gestire i profili delle credenziali su Windows con il modulo `AWSPowerShell` o con il modulo `AWSPowerShell.NetCore`.

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

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

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

Set-AWSCredential -ProfileName myProfileName
```

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

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

I seguenti esempi mostrano il comportamento del modulo `AWSPowerShell.NetCore` sui sistemi operativi Linux o macOS.

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

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

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

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

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

Set-AWSCredential -ProfileName myProfileName

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

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

Credenziali condivise negli AWS Tools for PowerShell

I Tools for Windows PowerShell supportano l'utilizzo del file delle credenziali AWS condivise in modo analogo all'AWS CLI e ad altri SDK AWS. I Tools for Windows PowerShell supportano ora la lettura e la scrittura dei profili delle credenziali `basic`, `session` e `assume role` sia nel file delle credenziali `.NET` che nel file delle credenziali AWS condivise. Questa funzionalità è abilitata da un nuovo namespace `Amazon.Runtime.CredentialManagement`.

⚠ Warning

Per evitare rischi per la sicurezza, non utilizzare gli utenti IAM per l'autenticazione quando sviluppi software creato ad hoc o lavori con dati reali. Utilizza invece la federazione con un provider di identità come [AWS IAM Identity Center](#).

ℹ Note

Le informazioni contenute in questo argomento riguardano le circostanze in cui è necessario ottenere e gestire manualmente le credenziali a breve o lungo termine. Per ulteriori informazioni sulle credenziali a breve e lungo termine, consulta l'argomento relativo agli [altri modi di autenticazione disponibili](#) nella Guida di riferimento per SDK e strumenti AWS. Per le best practice relative alla sicurezza, usa AWS IAM Identity Center, come descritto in [Configurazione dell'autenticazione degli strumenti](#).

I nuovi tipi di profilo e l'accesso al file delle credenziali AWS condiviso sono supportati dai seguenti parametri che sono stati aggiunti ai cmdlet relativi alle credenziali, [Initialize-AWSDefaultConfiguration](#), [New-AWSCredential](#) e [Set-AWSCredential](#). Nei cmdlet del servizio, puoi fare riferimento ai nuovi profili aggiungendo il parametro comune `-ProfileName`.

Utilizzo di un ruolo IAM con gli AWS Tools for PowerShell

Il file delle credenziali AWS condiviso consente ulteriori tipi di accesso. Ad esempio, puoi accedere alle tue risorse AWS utilizzando un ruolo IAM invece delle credenziali a lungo termine di un utente IAM. A tale scopo, devi disporre di un profilo standard con autorizzazioni per assumere il ruolo. Quando indichi a AWS Tools for PowerShell di utilizzare un profilo in cui è specificato un ruolo, AWS Tools for PowerShell cerca il profilo identificato dal parametro `SourceProfile`. Tali credenziali vengono utilizzate per richiedere credenziali temporanee per il ruolo specificato dal parametro `RoleArn`. Puoi richiedere l'utilizzo di un dispositivo di autenticazione a più fattori (Multi-Factor Authentication (MFA)) o di un codice `ExternalId` quando terze parti assumono il ruolo.

Nome parametro	Descrizione
<code>ExternalId</code>	L'ID esterno definito dall'utente da utilizzare quando assume un ruolo, se richiesto dal ruolo. In genere, è necessario solo quando deleghi

Nome parametro	Descrizione
	l'accesso al tuo account a terze parti. Le terze parti devono includere ExternalId come parametro quando assumono il ruolo assegnato . Per ulteriori informazioni, consulta la sezione Come utilizzare un ID esterno quando si concede l'accesso alle proprie risorse AWS a una terza parte nella Guida per l'utente IAM.
MfaSerial	Il numero seriale MFA da utilizzare quando assume un ruolo, se richiesto dal ruolo. Per ulteriori informazioni, consulta la sezione Utilizzo dell'autenticazione a più fattori (MFA) in AWS nella Guida per l'utente IAM.
RoleArn	L'ARN del ruolo da assumere per assumere le credenziali di ruolo. Per ulteriori informazioni sulla creazione e sull'utilizzo dei ruoli IAM, consulta la sezione relativa ai ruoli IAM nella Guida per l'utente IAM.
SourceProfile	Il nome del profilo di origine da utilizzare assumendo credenziali di ruolo. Le credenziali trovate in questo profilo vengono utilizzate per assumere il ruolo specificato dal parametro RoleArn.

Configurazione dei profili per assumere un ruolo

Di seguito è riportato un esempio che mostra come configurare un profilo fonte che consente di assumere direttamente un ruolo IAM.

Il primo comando crea un profilo di origine a cui fa riferimento il profilo del ruolo. Il secondo comando crea il profilo che quel ruolo assumerà. Il terzo comando mostra le credenziali per il profilo del ruolo.

```
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
```

Per utilizzare questo profilo del ruolo con i cmdlet del servizio Tools for Windows PowerShell, aggiungi il parametro comune `-ProfileName` al comando per fare riferimento al profilo del ruolo. Nell'esempio seguente viene utilizzato il profilo del ruolo definito nell'esempio precedente per accedere al cmdlet [Get-S3Bucket](#). AWS Tools for PowerShell cerca le credenziali in `my_source_profile`, utilizza tali credenziali per chiamare `AssumeRole` per conto dell'utente e quindi utilizza tali credenziali del ruolo temporanee per chiamare `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
```

Utilizzo dei tipi di credenziali del profilo

Per impostare il tipo di credenziali del profilo, comprendere quali parametri forniscono le informazioni necessarie per il tipo di profilo.

Tipo di credenziali	Parametri da utilizzare
Base	<code>-AccessKey</code>
Queste sono le credenziali a lungo termine per un utente IAM	<code>-SecretKey</code>
Sessione:	<code>-AccessKey</code> <code>-SecretKey</code>

Tipo di credenziali	Parametri da utilizzare
Queste sono le credenziali a breve termine per un ruolo IAM che puoi recuperare manualmente, ad esempio chiamando direttamente il cmdlet Use-STSRole .	-SessionToken
Ruolo:	-SourceProfile
Queste sono credenziali a breve termine per un ruolo IAM che gli AWS Tools for PowerShell recuperano per tuo conto.	-RoleArn facoltativo: -ExternalId facoltativo: -MfaSerial

Il parametro comune **ProfilesLocation**

È possibile utilizzare `-ProfileLocation` per scrivere nel file delle credenziali condivise e istruire un cmdlet per leggere il file delle credenziali. Aggiungendo il parametro `-ProfileLocation` è possibile controllare se i Tools for Windows PowerShell utilizzano il file delle credenziali condivise o il file delle credenziali .NET. La tabella seguente descrive come funziona il parametro nei Tools for Windows PowerShell.

Valore posizione profilo	Comportamento della risoluzione del profilo
null (non impostato) o vuoto	In primo luogo, cerca il file delle credenziali .NET per un profilo con il nome specificato. Se non è possibile trovare il profilo, cerca il file delle credenziali AWS condiviso in <i>(user's home directory) \.aws\credentials</i> .
Il percorso di un file nel formato di file delle credenziali AWS condiviso	Cerca solo un profilo con uno specifico nome nel file specificato.

Salvare le credenziali in un file delle credenziali

Per scrivere e salvare le credenziali in uno dei due file delle credenziali, eseguire il cmdlet `Set-AWSCredential`. Gli esempi seguenti mostrano come fare: Il primo comando utilizza `Set-`

`AWSCredential` con `-ProfileLocation` per aggiungere chiavi segrete e di accesso a un profilo specificato dal parametro `-ProfileName`. Nella seconda riga, eseguire il cmdlet [Get-Content](#) per visualizzare il contenuto del file delle credenziali.

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

aws_access_key_id=access_key2
aws_secret_access_key=secret_key2
```

Visualizzazione dei profili delle credenziali

Esegui il cmdlet [Get-AWSCredential](#) e aggiungi il parametro `-ListProfileDetail` per restituire i percorsi, i tipi di file delle credenziali e un elenco di nomi di profilo.

```
PS > Get-AWSCredential -ListProfileDetail

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

Rimozione dei profili delle credenziali

Per rimuovere i profili delle credenziali, esegui il nuovo cmdlet [Remove-AWSCredentialProfile](#). [Clear-AWSCredential](#) è obsoleto, ma ancora disponibile per la compatibilità con le versioni precedenti.

Note importanti

Solo [Initialize-AWSDefaultConfiguration](#), [New-AWSCredential](#) e [Set-AWSCredential](#) supportano i parametri per i profili dei ruoli. Non puoi specificare i parametri del ruolo direttamente in un comando come `Get-S3Bucket -SourceProfile source_profile_name -RoleArn arn:aws:iam::999999999999:role/role_name`. Ciò non funziona perché i cmdlet di servizio non supportano direttamente i parametri `SourceProfile` o `RoleArn`. Devi invece archiviare tali parametri in un profilo, quindi chiamare il comando con il parametro `-ProfileName`.

Utilizzo dei servizi AWS in AWS Tools for PowerShell

Questa sezione fornisce esempi di utilizzo degli AWS Tools for PowerShell per accedere ai servizi AWS. Questi esempi hanno lo scopo di dimostrare come utilizzare i cmdlets per eseguire processi reali in AWS. Questi esempi si basano sui cmdlet forniti da Tools for PowerShell. Per vedere quali cmdlet sono disponibili, consulta [AWS Tools for PowerShell Cmdlet Reference](#).

Codifica concatenazione dei file PowerShell

Alcuni cmdlet negli AWS Tools for PowerShell modificano i file o i record esistenti in AWS. Un esempio è `Edit-R53ResourceRecordSet`, che chiama l'API [ChangeResourceRecordSets](#) per Amazon Route 53.

Quando si modificano o si concatenano in file in PowerShell 5.1 o versioni precedenti, PowerShell codifica l'output in UTF-16, non UTF-8. In questo modo si possono aggiungere caratteri indesiderati e creare risultati non validi. Un editor esadecimale è in grado di visualizzare i caratteri indesiderati.

Per evitare la conversione di file di output in UTF-16, puoi reindirizzare il comando nel cmdlet `Out-File` di PowerShell e specificare una codifica UTF-8, come mostrato nel seguente esempio.

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

Se esegui i comandi AWS CLI dalla console PowerShell, viene applicato lo stesso comportamento. Puoi indirizzare l'output di un comando AWS CLI a `Out-File` nella console PowerShell. Altri cmdlet, ad esempio `Export-Csv` o `Export-Clixml`, inoltre, dispongono di un parametro `Encoding`. Per un elenco completo dei cmdlet che dispongono di un parametro `Encoding` e che consentono di correggere la codifica di un file concatenato di output, esegui il comando seguente:

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

Note

PowerShell 6.0 e versioni più recenti, incluso PowerShell Core, mantiene automaticamente la codifica UTF-8 per l'output di file concatenato.

Oggetti restituiti per gli strumenti PowerShell

Per rendere gli AWS Tools for PowerShell più utili in un ambiente PowerShell nativo, l'oggetto restituito da un cmdlet AWS Tools for PowerShell è un oggetto .NET, non l'oggetto di testo JSON che viene in genere restituito dall'API corrispondente nell'SDK AWS. Ad esempio, `Get-S3Bucket` restituisce una raccolta `Buckets`, non un oggetto di risposta JSON Amazon S3. Puoi inserire la raccolta `Buckets` nella pipeline PowerShell e interagire con essa nei modi appropriati. Allo stesso modo, `Get-EC2Instance` emette una raccolta di oggetti .NET `Reservation`, non un oggetto risultato JSON `DescribeEC2Instances`. Questo comportamento è impostato in maniera predefinita e consente di creare un'esperienza degli AWS Tools for PowerShell più coerente con la PowerShell idiomatica.

Le risposte effettive del servizio sono disponibili se ne hai bisogno. Sono archiviate come proprietà note sugli oggetti restituiti. Per le azioni API che supportano i campi `NextToken`, queste sono anche collegate come proprietà note.

Amazon EC2

Questa sezione esamina le fasi necessarie per avviare un'istanza Amazon EC2, ad esempio come:

- Recuperare un elenco delle Amazon Machine Images (AMI).
- Creare una coppia di chiavi per l'autenticazione SSH.
- Creare e configurare un gruppo di sicurezza Amazon EC2.
- Avviare l'istanza e recuperare le relative informazioni.

Amazon S3

La sezione esamina le fasi necessarie per creare un website statico ospitato in Amazon S3. Dimostra come fare a:

- Creare ed eliminare bucket Amazon S3.
- Caricare file in un bucket Amazon S3 come oggetti.
- Eliminare gli oggetti da un bucket Amazon S3.
- Designare un bucket Amazon S3 come website.

[AWS Lambda e AWS Tools for PowerShell](#)

Questa sezione fornisce una breve panoramica del modulo AWS Lambda Tools for PowerShell e descrive i passaggi per impostare il modulo.

[Amazon SNS e Amazon SQS](#)

Questa sezione esamina le fasi necessarie per sottoscrivere una coda Amazon SQS a un argomento Amazon SNS. Dimostra come fare a:

- Creazione di un argomento Amazon SNS.
- Creare una coda Amazon SQS.
- Sottoscrivere la coda all'argomento .
- Inviare un messaggio all'argomento.
- Ricevere il messaggio dalla coda.

[CloudWatch](#)

Questa sezione mostra un esempio di come pubblicare dati personalizzati su CloudWatch.

- Pubblicare un parametro personalizzato nel pannello di controllo CloudWatch.

Vedi anche

- [Nozioni di base su AWS Tools for Windows PowerShell](#)

Argomenti

- [Amazon S3 e Tools for Windows PowerShell](#)
- [Amazon EC2 e Tools for Windows PowerShell](#)
- [AWS Lambda e AWS Tools for PowerShell](#)
- [Amazon SQS, Amazon SNS e Tools for Windows PowerShell](#)
- [Utilizzo degli AWS Tools for Windows PowerShell per pubblicare in CloudWatch](#)

- [Uso del parametro ClientConfig nei cmdlet](#)

Amazon S3 e Tools for Windows PowerShell

In questa sezione, creiamo un website statico con gli AWS Tools for Windows PowerShell utilizzando Amazon S3 e CloudFront. Durante il processo, dimostriamo una serie di operazioni comuni con questi servizi. Questa spiegazione passo per passo è modellata sulla base della Guida introduttiva per l'[Hosting di un website statico](#), in cui viene descritto un processo analogo utilizzando la [Console di gestione AWS](#).

I comandi mostrati qui presuppongono che siano state impostate le credenziali e una regione predefinite per la sessione PowerShell. Pertanto, le credenziali e le regioni non sono incluse nelle invocazioni dei cmdlet.

Note

Attualmente non esiste un'API Amazon S3 per rinominare un bucket o un oggetto e, pertanto, non esiste un singolo cmdlet Tools for Windows PowerShell per eseguire questo processo. Per rinominare un oggetto in S3, consigliamo di copiare l'oggetto su uno con un nuovo nome, eseguendo il cmdlet [Copy-S3Object](#), quindi eliminando l'oggetto originale eseguendo il cmdlet [Remove-S3Object](#).

Consulta anche

- [Utilizzo dei servizi AWS in AWS Tools for PowerShell](#)
- [Hosting di un sito Web statico su Amazon S3](#)
- [Console Amazon S3](#)

Argomenti

- [Creare un bucket Amazon S3, verificare la sua Regione e rimuoverlo se si desidera](#)
- [Configurare un bucket Amazon S3 come website e abilitare la registrazione](#)
- [Caricare oggetti in un bucket Amazon S3](#)
- [Eliminare bucket e oggetti Amazon S3](#)
- [Caricamento dei contenuti di testo in linea su Amazon S3](#)

Creare un bucket Amazon S3, verificare la sua Regione e rimuoverlo se si desidera

Utilizza il cmdlet `New-S3Bucket` per creare un nuovo bucket Amazon S3. Negli esempi seguenti viene creato un bucket denominato `website-example`. Il nome del bucket deve essere univoco in tutte le regioni. Nell'esempio viene creato il bucket nella regione `us-west-1`.

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

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

È possibile verificare la regione in cui si trova il bucket utilizzando il cmdlet `Get-S3BucketLocation`.

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

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

Al termine del tutorial, puoi usare la riga seguente per rimuovere questo bucket. Sugeriamo di lasciare questo bucket al suo posto visto che lo utilizziamo negli esempi successivi.

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

Il completamento del processo di rimozione del bucket potrebbe richiedere tempo. Se provi a ricreare immediatamente un bucket con lo stesso nome, il cmdlet `New-S3Bucket` può non avere esito positivo fino a quando il vecchio non è completamente eliminato.

Vedi anche

- [Utilizzo dei servizi AWS in AWS Tools for PowerShell](#)
- [Inserimento del bucket \(Amazon S3 Service Reference\)](#)
- [AWSRegioni AWS PowerShell per Amazon S3](#)

Configurare un bucket Amazon S3 come website e abilitare la registrazione

Utilizza il cmdlet `Write-S3BucketWebsite` per configurare un bucket Amazon S3 come website statico. L'esempio seguente specifica un nome `index.html` per la pagina Web di contenuti e un nome `error.html` per la pagina Web di errore predefinita. Questo cmdlet non consente di creare tali pagine. Devono essere [caricate come oggetti di Amazon S3](#).

```
PS > Write-S3BucketWebsite -BucketName website-example -  
WebsiteConfiguration_IndexDocumentSuffix index.html -WebsiteConfiguration_ErrorDocument  
error.html  
RequestId      : A1813E27995FFDDD  
AmazonId2      : T7h1D0eLqA5Q2XfTe8j2q3SLoP3/5XwhUU3RyJBGHU/LnC+CIWLeGgP0MY24xA1I  
ResponseStream :  
Headers        : {x-amz-id-2, x-amz-request-id, Content-Length, Date...}  
Metadata       : {}  
ResponseXml    :
```

Vedi anche

- [Utilizzo dei servizi AWS in AWS Tools for PowerShell](#)
- [Inserire sito Web bucket \(Amazon S3 API Reference\)](#)
- [Inserire Bucket ACL \(Amazon S3 API Reference\)](#)

Caricare oggetti in un bucket Amazon S3

Utilizza il cmdlet `Write-S3Object` per caricare file dal file system locale in un bucket Amazon S3 come oggetti. L'esempio seguente crea e carica due semplici file HTML in un bucket Amazon S3 e verifica che gli oggetti siano stati caricati. Il parametro `-File` in `Write-S3Object` specifica il nome del file nel file system locale. Il parametro `-Key` specifica il nome che avrà l'oggetto corrispondente in Amazon S3.

Amazon deduce automaticamente il tipo di contenuto degli oggetti dalle estensioni dei file, in questo caso, ".html".

```
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

```

Opzioni di ACL predefinite

I valori per specificare le liste di controllo accessi (ACL) predefinite con i Tools for Windows PowerShell sono gli stessi utilizzati dall' AWS SDK for .NET. Tieni presente, tuttavia, che questi sono diversi dai valori utilizzati dall'operazione `Amazon S3Put Object`. I Tools for Windows PowerShell supportano le seguenti liste di controllo accessi (ACL) predefinite:

- NoACL
- private
- public-read
- public-read-write

- `aws-exec-read`
- `authenticated-read`
- `bucket-owner-read`
- `bucket-owner-full-control`
- `log-delivery-write`

Per ulteriori informazioni su queste impostazioni di ACL predefinite, consultare la pagina [Panoramica lista di controllo accessi \(ACL\)](#).

Nota riguardo il caricamento in più parti

Se si utilizza l'API di Amazon S3 per caricare un file di dimensioni superiori a 5 GB, è necessario utilizzare il caricamento in più parti. Tuttavia, il cmdlet `Write-S3Object` fornito dai Tools for Windows PowerShell è in grado di gestire in modo trasparente il caricamento di file di dimensioni superiori a 5 GB.

Test del sito Web

A questo punto, è possibile testare il sito Web navigando con un browser. Gli URL per i website statici ospitati in Amazon S3 seguono un formato standard.

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

Ad esempio:

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

Vedi anche

- [Utilizzo dei servizi AWS in AWS Tools for PowerShell](#)
- [Inserisci oggetto \(Amazon S3 API Reference\)](#)
- [ACL predefinite \(Amazon S3 API Reference\)](#)

Eliminare bucket e oggetti Amazon S3

Questa sezione descrive come eliminare il sito Web creato nelle sezioni precedenti. È possibile semplicemente eliminare gli oggetti per i file HTML e quindi eliminare il bucket Amazon S3 per il sito.

Innanzitutto, esegui il cmdlet `Remove-S3Object` per eliminare gli oggetti per i file HTML dal bucket Amazon S3.

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

La risposta `False` è un artefatto previsto del modo in cui Amazon S3 elabora la richiesta. In questo contesto, non indica un problema.

Ora puoi eseguire il cmdlet `Remove-S3Bucket` per eliminare il bucket Amazon S3 ormai vuoto per il sito.

```
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    :
```

Nella versione 1.1 e nelle versioni più recenti degli AWS Tools for PowerShell, puoi aggiungere il parametro `-DeleteBucketContent` a `Remove-S3Bucket` che, per prima cosa, elimina tutti gli oggetti e tutte le versioni nel bucket specificato prima di provare a rimuovere il bucket stesso. A seconda del numero degli oggetti o delle versioni degli oggetti nel bucket, questa operazione potrebbe richiedere una notevole quantità di tempo. Nelle versioni dei Tools for Windows PowerShell precedenti alla 1.1, il bucket doveva essere vuoto affinché `Remove-S3Bucket` lo eliminasse.

Note

A meno che tu non aggiunga il parametro `-Force`, AWS Tools for PowerShell richiede la conferma prima dell'esecuzione del cmdlet.

Vedi anche

- [Utilizzo dei servizi AWS in AWS Tools for PowerShell](#)
- [Eliminazione di un oggetto \(Amazon S3 API Reference\)](#)
- [Eliminazione del bucket \(Amazon S3 API Reference\)](#)

Caricamento dei contenuti di testo in linea su Amazon S3

Il cmdlet `Write-S3Object` supporta la possibilità di caricare contenuti di testo in linea su Amazon S3. Utilizzando il parametro `-Content` (alias `-Text`), è possibile specificare contenuti basati su testo che devono essere caricati su Amazon S3 senza la necessità di trasferirli prima in un file. Il parametro accetta stringhe semplici di una sola riga e anche here strings che contengono più righe.

```
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 e Tools for Windows PowerShell

Puoi eseguire processi comuni relativi ad Amazon EC2 utilizzando gli AWS Tools for PowerShell.

I comandi mostrati qui presuppongono che siano state impostate le credenziali e la regione predefinite per la sessione PowerShell. Pertanto, non includiamo le credenziali o la regione quando

invochiamo il cmdlet. Per ulteriori informazioni, consulta [. Nozioni di base su AWS Tools for Windows PowerShell.](#)

Argomenti

- [Creazione di una coppia di chiavi](#)
- [Creare un gruppo di sicurezza con Windows PowerShell](#)
- [Trovare una Amazon Machine Image con Windows PowerShell](#)
- [Avvia un'istanza Amazon EC2 con Windows PowerShell](#)

Creazione di una coppia di chiavi

Il seguente esempio `New-EC2KeyPair` crea una coppia di chiavi e archivia nella variabile `$myPSKeyPair` PowerShell

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

Indirizza l'oggetto della coppia di chiavi nel cmdlet `Get-Member` per visualizzare la struttura dell'oggetto.

```
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;}

Indirizzare l'oggetto della coppia di chiavi nel cmdlet `Format-List` per visualizzare i valori di `KeyName`, `KeyFingerprint` e i membri di `KeyMaterial`. (L'output è stato accorciato per ragioni di leggibilità).

```
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----
                  MIIIEogIBAAKCAQEAKK+ANYUS9c7niNjYfaCn6KYj/D0I6djnFoQE...
                  Mz6btocPcE7EMeH1wySUP8nouAS9xbl9L7+Vkd74bN9KmNcPa/Mu...
                  Zyn4vVe0Q5il/MpkrRogHq0B0rigeTeV5Yc3lv00RFFPu0Kz4kcm...
                  w3Jg8dKsWn0p10pX7V3sRC02KgJIbejQUvBFGi50QK9bm4tXBIEc...
                  daxKIAQMtDUdmBDrhR1/YMv8itFe5DiLLbq7Ga+FDcS85NstBa3h...
                  iuskGkcvGwKcFQkLmRHRoDpPb+0dFsZtjHZDpMVFmA9tT8EdbkEF...
                  3SrNeqZPsxJJIX0odb3CxLJpg75JU5kyWnb0+sDNVHoJiZCULCr0...
                  GGLLfEgB95KjGIk7zEv2Q7K6s+DHclrDeMZWa7KFNRZuCuX7jssC...
                  x098abxMr3o3TNU6p1ZYRJEQ0oJr0W+kc+/8SWb8NIwfltwmJEy...
                  1BX9X8WFX/A8VLHrT1e1rKmlkNECgYEAwltkV1p0JAFhz9p7ZFEv...
                  vvVsPaF0Ev9bk9pqhx269PB50x2KokwCagDMMaYvasWobuLmNu/1...
                  lmwRx7KTeQ7W1J30LgxHA1QNMkip9c4Tb3q9vVc3t/fPf8vwfJ8C...
                  63g6N6rk2FkHZX1E62BgbewUd3eZ0S05Ip4VUdvtGcuc8/qa+e5C...
                  KXgyt9n164pMv+VaXfXkZhdLAdY0Khc9TGB9++VMSG5TrD15YJId...
                  gYALEI7m1jJKpHWAes0hiemw5VmKyIZpzGstSJsFStER1AjiETDH...
                  YAtnI4J8dRyP9I7B0V0n3wNfIjk85gi1/00c+j8S65giLafndWGR...
                  9R9wIkm5BMUcSRRcDy0yuwKBgEbkOnGGSD0ah4HkvrUkepIbUDTD...
                  AnEBM1cXI5UT7BfKInpUihZi59QhgdK/hk0SmWhlZGWikJ5VizBf...
                  drkBr/vTKVRMTi31VFB7KkIV1xJxC5E/BZ+YdZEpWoCZAoGAC/Cd...
                  TTld5N6opg0XAcQJwzqoGa9ZMwc5Q9f4bfRc67emkw0ZAAwSsvWR...
                  x302duuy7/smTwWwskEWRK5IrUxoMv/VVYaqdzc0ajwieNrb1r7c...
                  -----END RSA PRIVATE KEY-----

```

Il membro `KeyMaterial` archivia la chiave privata per la coppia di chiavi. La chiave pubblica viene archiviata in AWS. Non è possibile recuperare la chiave pubblica da AWS, ma è possibile verificare la chiave pubblica confrontando il `KeyFingerprint` per la chiave privata con quello restituito da AWS per la chiave pubblica.

Visualizzazione dell'impronta della coppia di chiave

È possibile utilizzare il cmdlet `Get-EC2KeyPair` per visualizzare l'impronta per la coppia di chiavi.

```

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

```

Archiviazione della chiave privata

Per archiviare la chiave privata in un file, indirizzare il membro `KeyFingerMaterial` al cmdlet `Out-File`.

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

È necessario specificare `-Encoding ascii` quando si scrive la chiave privata per un file. In caso contrario, strumenti come `openssl` potrebbero non essere in grado di leggere il file correttamente. È possibile verificare la correttezza del formato del file risultante utilizzando un comando, ad esempio il seguente:

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

Lo strumento `openssl` non è incluso negli AWS Tools for PowerShell o nell'AWS SDK for .NET.

Rimozione di una coppia di chiavi

Per avviare un'istanza e connetterti a essa è necessaria una coppia di chiavi. Al termine dell'utilizzo di una coppia di chiavi, puoi rimuoverla. Per rimuovere la chiave pubblica da AWS, utilizza il cmdlet `Remove-EC2KeyPair`. Quando richiesto, premi `Enter` per rimuovere la coppia di chiavi.

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

Confirm

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

La variabile, `$myPSKeyPair`, è ancora presente nella sessione PowerShell corrente e contiene ancora le informazioni sulla coppia di chiavi. Anche il file `myPSKeyPair.pem` esiste. Tuttavia, la chiave privata non è più valida perché la chiave pubblica per la coppia di chiavi non è più archiviata in AWS.

Creare un gruppo di sicurezza con Windows PowerShell

Puoi utilizzare gli AWS Tools for PowerShell per creare e configurare un gruppo di sicurezza. Quando crei un gruppo di sicurezza, devi specificare se è per EC2-Classical o EC2-VPC. La risposta è l'ID del gruppo di sicurezza.

Se hai bisogno di connetterti a un'istanza, devi configurare il gruppo di sicurezza per consentire il traffico SSH (Linux) o il traffico RDP (Windows).

Argomenti

- [Prerequisiti](#)
- [Creazione di un gruppo di sicurezza per EC2-Classical](#)
- [Creazione di un gruppo di sicurezza per EC2-VPC](#)

Prerequisiti

È necessario l'indirizzo IP pubblico del computer, nella notazione CIDR. Puoi ottenere l'indirizzo IP pubblico del computer locale utilizzando un servizio. Ad esempio, Amazon fornisce il servizio seguente: <http://checkip.amazonaws.com/> o <https://checkip.amazonaws.com/>. Per individuare un altro servizio che fornisca l'indirizzo IP, usa la frase di ricerca "qual è il mio indirizzo IP". Se ti stai connettendo tramite un ISP o con la protezione di un firewall senza un indirizzo IP statico, devi individuare l'intervallo degli indirizzi IP che i computer client possono utilizzare.

Warning

Se specifichi `0.0.0.0/0`, abiliti il traffico da qualsiasi indirizzo IP nel mondo. Per i protocolli SSH e RDP, potresti considerarlo accettabile per un breve periodo di tempo in un ambiente di test, ma non è sicuro per gli ambienti di produzione. In produzione, assicurati di autorizzare l'accesso solo dall'indirizzo IP individuale o dall'intervallo di indirizzi appropriato.

Creazione di un gruppo di sicurezza per EC2-Classical

Warning

Ritiremo EC2-Classical il 15 agosto 2022. Sugeriamo di effettuare la migrazione da EC2-Classical a un VPC. Per ulteriori informazioni, consulta [Eseguire la migrazione da EC2-Classical a un VPC](#) nella [Guida per l'utente di Amazon EC2 per istanze Linux](#) o nella [Guida per l'utente di Amazon EC2 per istanze Windows](#). Per ulteriori informazioni, consulta il post di blog [II networking EC2-Classical va in pensione: ecco come prepararsi](#).

L'esempio seguente usa il cmdlet `New-EC2SecurityGroup` per creare un gruppo di sicurezza per `EC2-Classic`.

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

```
sg-0a346530123456789
```

Per visualizzare la configurazione iniziale del gruppo di sicurezza, utilizza il 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
```

Per configurare il gruppo di sicurezza per consentire il traffico in entrata sulla porta TCP 22 (SSH) e sulla porta TCP 3389, utilizzare il cmdlet `Grant-EC2SecurityGroupIngress`. Ad esempio, lo script di esempio seguente mostra come abilitare il traffico SSH da un singolo indirizzo IP, `203.0.113.25/32`.

```
$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
```

Per verificare che il gruppo di sicurezza sia stato aggiornato, esegui di nuovo il cmdlet `Get-EC2SecurityGroup`. Non puoi specificare una regola in uscita per `EC2-Classic`.

```
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             : {}
```

Per visualizzare la regola del gruppo di sicurezza, utilizza la proprietà `IpPermissions`.

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

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

Creazione di un gruppo di sicurezza per EC2-VPC

L'esempio `New-EC2SecurityGroup` seguente aggiunge il parametro `-VpcId` per creare un gruppo di sicurezza per il VPC specificato.

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

Per visualizzare la configurazione iniziale del gruppo di sicurezza, utilizza il cmdlet `Get-EC2SecurityGroup`. Per impostazione predefinita, il gruppo di sicurezza per un VPC contiene una regola che abilita tutto il traffico in uscita. Notare che non è possibile fare riferimento a un gruppo di sicurezza per EC2-VPC utilizzando il nome.

```
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                 : {}
```

Per definire le autorizzazioni per il traffico in entrata sulla porta TCP 22 (SSH) e sulla porta TCP 3389, utilizza il cmdlet `New-Object`. Lo script di esempio seguente definisce le autorizzazioni per le porte TCP 22 e 3389 da un singolo indirizzo IP, `203.0.113.25/32`.

```
$ip1 = new-object Amazon.EC2.Model.IpPermission
$ip1.IpProtocol = "tcp"
$ip1.FromPort = 22
$ip1.ToPort = 22
$ip1.IpRanges.Add("203.0.113.25/32")
$ip2 = new-object Amazon.EC2.Model.IpPermission
$ip2.IpProtocol = "tcp"
$ip2.FromPort = 3389
$ip2.ToPort = 3389
$ip2.IpRanges.Add("203.0.113.25/32")
Grant-EC2SecurityGroupIngress -GroupId $groupid -IpPermissions @( $ip1, $ip2 )
```

Per verificare che il gruppo di sicurezza sia stato aggiornato, utilizza di nuovo il cmdlet `Get-EC2SecurityGroup`.

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

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

Per visualizzare le regole in entrata, puoi recuperare la proprietà `IpPermissions` dall'oggetto raccolto restituito dal comando precedente.

```
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}
```

Trovare una Amazon Machine Image con Windows PowerShell

Quando si avvia un'istanza Amazon EC2, è necessario specificare un'Amazon Machine Image (AMI) da utilizzare come modello per l'istanza. Tuttavia, gli ID per le AMI AWS Windows cambiano frequentemente perché AWS fornisce nuove AMI con i più recenti aggiornamenti e miglioramenti della sicurezza. Puoi utilizzare i cmdlet [Get-EC2Image](#) e [Get-EC2ImageByName](#) per trovare le AMI di Windows correnti e ottenere i loro ID.

Argomenti

- [Get-EC2Image](#)
- [Get-EC2ImageByName](#)

Get-EC2Image

Il cmdlet `Get-EC2Image` recupera un elenco di AMI che è possibile utilizzare.

Utilizza il parametro `-Owner` con il valore dell'array `amazon, self` in modo che `Get-EC2Image` recuperi solo le AMI che appartengono ad Amazon o a te. In questo contesto, devi fare riferimento all'utente di cui hai usato le credenziali per richiamare il cmdlet.

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

È possibile definire i risultati utilizzando il parametro `-Filter`. Per specificare il filtro, creare un oggetto di tipo `Amazon.EC2.Model.Filter`. Ad esempio, utilizzare il filtro seguente per visualizzare solo le AMI Windows.

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

```
Get-EC2Image -Owner amazon, self -Filter $filter_platform
```

Di seguito è riportato un esempio di una delle AMI restituite dal cmdlet; l'output effettivo del comando precedente fornisce informazioni per molte AMI.

```
Architecture      : x86_64
BlockDeviceMappings : {/dev/sda1, xvdca, xvdcb, xvdcc...}
CreationDate      : 2019-06-12T10:41:31.000Z
Description       : Microsoft Windows Server 2019 Full Locale English with SQL Web
                   2017 AMI provided by Amazon
EnaSupport        : True
Hypervisor        : xen
ImageId           : ami-000226b77608d973b
ImageLocation     : amazon/Windows_Server-2019-English-Full-SQL_2017_Web-2019.06.12
ImageOwnerAlias   : amazon
ImageType         : machine
KernelId         :
Name              : Windows_Server-2019-English-Full-SQL_2017_Web-2019.06.12
OwnerId          : 801119661308
Platform         : Windows
ProductCodes     : {}
Public           : True
RamdiskId        :
RootDeviceName    : /dev/sda1
RootDeviceType   : ebs
SriovNetSupport  : simple
State            : available
StateReason      :
Tags             : {}
VirtualizationType : hvm
```

Get-EC2ImageByName

Il cmdlet `Get-EC2ImageByName` consente di filtrare l'elenco di AMI AWS Windows in base al tipo di configurazione del server interessato.

Quando viene eseguito senza parametri, come di seguito, il cmdlet emette il set completo di nomi di filtro corrente:

```
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
```

Per restringere il set di immagini restituito, specificare uno o più nomi di filtri utilizzando il parametro `Names`.

```
PS > Get-EC2ImageByName -Names WINDOWS_2016_CORE
```

```
Architecture      : x86_64
BlockDeviceMappings : {/dev/sda1, xvdca, xvdcb, xvdcc...}
CreationDate      : 2019-08-16T09:36:09.000Z
Description       : Microsoft Windows Server 2016 Core Locale English AMI provided by
  Amazon
EnaSupport        : True
Hypervisor        : xen
ImageId           : ami-06f2a2afca06f15fc
ImageLocation     : amazon/Windows_Server-2016-English-Core-Base-2019.08.16
ImageOwnerAlias   : amazon
ImageType         : machine
KernelId          :
Name              : Windows_Server-2016-English-Core-Base-2019.08.16
OwnerId           : 801119661308
Platform          : Windows
ProductCodes      : {}
Public            : True
RamdiskId         :
RootDeviceName    : /dev/sda1
RootDeviceType    : ebs
SriovNetSupport   : simple
State             : available
StateReason       :
Tags              : {}
VirtualizationType : hvm
```

Avvia un'istanza Amazon EC2 con Windows PowerShell

Per avviare un'istanza di Amazon EC2, sono necessari la coppia di chiavi e il gruppo di sicurezza creati nelle sezioni precedenti. È inoltre necessario l'ID di una Amazon Machine Image (AMI). Per ulteriori informazioni, consulta la seguente documentazione:

- [Creazione di una coppia di chiavi](#)
- [Creare un gruppo di sicurezza con Windows PowerShell](#)
- [Trovare una Amazon Machine Image con Windows PowerShell](#)

⚠ Important

Se avvii un'istanza che non rientra nel piano gratuito, i costi ti verranno addebitati dopo l'avvio dell'istanza e per il periodo di tempo in cui l'istanza viene eseguita, anche se rimane inattiva.

Argomenti

- [Avviare un'istanza in EC2-Classic](#)
- [Avviare un'istanza in una VPC](#)
- [Avviare un'istanza Spot in una VPC](#)

Avviare un'istanza in EC2-Classic**⚠ Warning**

Ritireremo EC2-Classic il 15 agosto 2022. Suggeriamo di effettuare la migrazione da EC2-Classic a un VPC. Per ulteriori informazioni, consulta [Eseguire la migrazione da EC2-Classic a un VPC nella Guida per l'utente di Amazon EC2 per istanze Linux](#) o nella [Guida per l'utente di Amazon EC2 per istanze Windows](#). Per ulteriori informazioni, consulta il post di blog [Il networking EC2-Classic va in pensione: ecco come prepararsi](#).

Il comando seguente crea e avvia un'istanza t1.micro singola.

```
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     : {}
```

Inizialmente l'istanza è nello stato `pending`, ma in pochi minuti passa allo stato `running`. Per visualizzare le informazioni sull'istanza, utilizza il cmdlet `Get-EC2Instance`. Se hai più di un'istanza, puoi filtrare i risultati dell'ID della prenotazione con il parametro `Filter`. Innanzitutto, devi creare un oggetto di tipo `Amazon.EC2.Model.Filter`. Successivamente, devi chiamare `Get-EC2Instance` che utilizza il filtro e quindi visualizza la proprietà `Instances`.

```
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            :
```

Avviare un'istanza in una VPC

Il comando seguente crea una singola istanza `m1.small` nella sottorete privata specificata. Il gruppo di sicurezza deve essere valido per la sottorete specificata.

```
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      : {}
```

Inizialmente l'istanza è nello stato `pending`, ma in pochi minuti passa allo stato `running`. Per visualizzare le informazioni sull'istanza, utilizza il cmdlet `Get-EC2Instance`. Se hai più di un'istanza, puoi filtrare i risultati dell'ID della prenotazione con il parametro `Filter`. Innanzitutto, devi creare un oggetto di tipo `Amazon.EC2.Model.Filter`. Successivamente, devi chiamare `Get-EC2Instance` che utilizza il filtro e quindi visualizza la proprietà `Instances`.

```
PS > $reservation = New-Object 'collections.generic.list[string]'  
PS > $reservation.add("r-b70a0ef1")  
PS > $filter_reservation = New-Object Amazon.EC2.Model.Filter -Property @{Name =  
  "reservation-id"; Values = $reservation}  
PS > (Get-EC2Instance -Filter $filter_reservation).Instances  
  
AmiLaunchIndex   : 0  
Architecture     : x86_64  
BlockDeviceMappings : {/dev/sda1}
```

```
ClientToken      :  
EbsOptimized    : False  
Hypervisor      : xen  
IamInstanceProfile :  
ImageId        : ami-c49c0dac  
InstanceId      : i-5203422c  
InstanceLifecycle :  
InstanceType   : m1.small  
KernelId       :  
KeyName        : myPSKeyPair  
LaunchTime     : 12/2/2018 3:38:52 PM  
Monitoring     : Amazon.EC2.Model.Monitoring  
NetworkInterfaces : {}  
Placement      : Amazon.EC2.Model.Placement  
Platform       : Windows  
PrivateDnsName :  
PrivateIpAddress : 10.25.1.11  
ProductCodes   : {}  
PublicDnsName  :  
PublicIpAddress : 198.51.100.245  
RamdiskId      :  
RootDeviceName : /dev/sda1  
RootDeviceType : ebs  
SecurityGroups : {myPSSecurityGroup}  
SourceDestCheck : True  
SpotInstanceRequestId :  
SriovNetSupport :  
State          : Amazon.EC2.Model.InstanceState  
StateReason    :  
StateTransitionReason :  
SubnetId      : subnet-d60013bf  
Tags          : {}  
VirtualizationType : hvm  
VpcId        : vpc-a01106c2
```

Avviare un'istanza Spot in una VPC

Il seguente script di esempio richiede un'istanza Spot nella sottorete specificata. Il gruppo di sicurezza deve essere uno creato per la VPC che contiene la sottorete specificata.

```
$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 e AWS Tools for PowerShell

Con il modulo [AWSLambdaPSCore](#), puoi sviluppare le funzioni AWS Lambda in PowerShell Core 6.0 utilizzando il runtime .NET Core 2.1. Gli sviluppatori PowerShell possono gestire le risorse AWS e scrivere script di automazione nell'ambiente PowerShell utilizzando Lambda. Il supporto PowerShell in Lambda consente di eseguire script PowerShell o funzioni Lambda in risposta a qualsiasi evento Lambda, ad esempio un evento pianificato Amazon S3 o Amazon CloudWatch. Il modulo AWSLambdaPSCore è un modulo AWS separato per PowerShell; non fa parte degli AWS Tools for PowerShell, né la sua installazione consente di installare gli AWS Tools for PowerShell.

Dopo aver installato il modulo AWSLambdaPSCore, è possibile utilizzare qualsiasi cmdlet PowerShell disponibile, o svilupparne di propri, per creare funzioni serverless. Il modulo AWS Lambda Tools for PowerShell include modelli di progetto per applicazioni serverless basate su PowerShell e strumenti per pubblicare progetti su AWS.

Il supporto per il modulo AWSLambdaPSCore è disponibile in tutte le Regioni che supportano Lambda. Per ulteriori informazioni sulle Regioni supportate, consulta la [tabella delle Regioni AWS](#).

Prerequisiti

I seguenti passaggi sono necessari prima di installare e utilizzare il modulo AWSLambdaPSCore. Per ulteriori informazioni su questi passaggi, vedere [Impostazione di un ambiente di sviluppo di PowerShell](#) nella Guida per lo sviluppatore AWS Lambda.

- Installare la versione corretta di PowerShell – il supporto di Lambda per PowerShell si basa sulla multiplatforma PowerShell Core versione 6.0. È possibile sviluppare funzioni Lambda in PowerShell su Windows PowerShell, Linux o Mac. Se non si dispone di questa versione di PowerShell installata, le istruzioni sono disponibili sul [Sito web della documentazione di Microsoft PowerShell](#).

- Installa l'SDK .NET Core 2.1: poiché PowerShell Core si basa su .NET Core, il supporto Lambda per PowerShell utilizza lo stesso runtime .NET Core 2.1 Lambda per le funzioni Lambda sviluppate sia in .NET Core che in PowerShell. I cmdlet di pubblicazione Lambda Powershell utilizzano l'SDK .NET Core 2.1 per creare il pacchetto di distribuzione Lambda. L'SDK .NET Core 2.1 SDK è disponibile nell'[Area download Microsoft](#). Assicurati di installare l'SDK, non il runtime.

Installa il modulo utilizzando AWSLambdaPSCore

Dopo aver completato i prerequisiti, è possibile installare il modulo AWSLambdaPSCore. In una sessione di PowerShell Core, esegui il comando seguente:

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

Sei pronto per iniziare a sviluppare funzioni Lambda in PowerShell. Per ulteriori informazioni su come iniziare, consulta [Modello di programmazione per la creazione delle funzioni Lambda in PowerShell](#) nella Guida per lo sviluppatoreAWS Lambda.

Vedi anche

- [Annuncio del supporto Lambda per PowerShell Core sul blog per gli sviluppatori AWS](#)
- [Modulo AWSLambdaPSCore sul sito web sulla galleria PowerShell](#)
- [Configurazione di un ambiente di sviluppo PowerShell](#)
- [AWS Lambda Tools for Powershell on GitHub](#)
- [Console AWS Lambda](#)

Amazon SQS, Amazon SNS e Tools for Windows PowerShell

In questa sezione vengono forniti esempi in cui viene illustrato come:

- Creare una coda Amazon SQS e ottenere l'ARN della coda (Amazon Resource Name).
- Creare un argomento Amazon SNS.
- Concedere l'autorizzazione all'argomento SNS, in modo che possa inviare messaggi alla coda.
- Sottoscrivere la coda all'argomento SNS
- Concedere agli utenti IAM o agli account AWS le autorizzazioni per pubblicare nell'argomento SNS e leggere i messaggi dalla coda SQS.

- Verificare i risultati pubblicando un messaggio nell'argomento e leggendo il messaggio dalla coda.

Creare una coda Amazon SQS e ottenere l'ARN della coda

Il comando seguente crea una coda SQS nella regione predefinita. L'output mostra l'URL della nuova coda.

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

Il comando seguente recupera l'ARN della coda.

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

Creare un argomento Amazon SNS.

Il comando seguente crea un argomento SNS nella regione predefinita e restituisce l'ARN del nuovo argomento.

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

Concedere le autorizzazioni all'argomento SNS

Il seguente script di esempio crea sia una coda SQS sia un argomento SNS e concede le autorizzazioni all'argomento SNS in modo che possa inviare messaggi alla coda SQS:

```
# create the queue and topic to be associated
$qurl = New-SQSQueue -QueueName "myQueue"
$topicarn = New-SNSTopic -Name "myTopic"

# get the queue ARN to inject into the policy; it will be returned
# in the output's QueueARN member but we need to put it into a variable
# so text expansion in the policy string takes effect
$qarn = (Get-SQSQueueAttribute -QueueUrl $qurl -AttributeNames "QueueArn").QueueARN
```

```
# construct the policy and inject arns
$policy = @"
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": "*",
    "Action": "SQS:SendMessage",
    "Resource": "$qarn",
    "Condition": { "ArnEquals": { "aws:SourceArn": "$topicarn" } }
  }
}
"@

# set the policy
Set-SQSQueueAttribute -QueueUrl $qurl -Attribute @{ Policy=$policy }
```

Sottoscrivere la coda all'argomento SNS

Il comando seguente sottoscrive la coda myQueue all'argomento SNS myTopic e restituisce l'ID sottoscrizione:

```
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
```

Concedere le autorizzazioni

Il comando seguente concede l'autorizzazione per eseguire l'azione sns:Publish sull'argomento myTopic.

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

Il comando seguente concede l'autorizzazione per eseguire le azioni sqs:ReceiveMessage e sqs>DeleteMessage sulla coda myQueue.

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

Verificare i risultati

Il comando seguente verifica la nuova coda e l'argomento pubblicando un messaggio nell'argomento SNS myTopic e restituisce MessageId.

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

Il comando seguente recupera il messaggio dalla coda SQS myQueue e lo mostra.

```
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" :
  "11E17A2+X0uJZnw3TlgcXz4C4KPLXZxbxoEMIirelh13u/oxkWmz5+9tJKFMns1Z0qVqKxk
+ExfEZcD5yWt6biVuBb8pyRmZ1b03hUENl3ayv2WQiQT1vpLpM7VEQN5m+hLIiPFcs
vyuGkJReV710JWPHnCN
+qTE2lId2RPkF0eGtLGawTsSPTWEvJdDbL1f7E0zZ0q1niXTUtpsZ8Swx01X3Q06u9i9qBFt0ekJFZNJp6Avu05hIklb4yoc
y0a8Y19lWp7a7EoWaBn0zhCESe7o
kZC6ncBJWphX7KCGVYD0qhVf/5VDgBuv9w8T+higJyv13WbaSvg==",
  "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      :
  AQEB2vvk1e5c0KFjeIWJticabkc664yuDEjhucnI0qdVUmie7bX7GiJb17F0enABUgaI2XjEcNPxixhVc/
wfsAJZLNHn18S1bQa0R/kD+Saaq40Ivfj8x3M40h1yM1cVKpYmhAzsYrAwAD5g5FvxNBD6zs
  +HmXdkax2Wd+9AxrH1QZV5ur1MoByKWWbDbsqoYJTJquCc10gWIak/sBx/
daBRMTiVQ4GHsrQWMVhtNC14q7Jy/0L2dkmb4dzJfJq0VbFSX1G+u/lrSLpgae+Dfux646y8yFiPFzY4ua4mCF/
SVUn63Spy
  sHN12776axknhg3j9K/Xwj54DixdsegrKoLx+ctI
+0jzAetBR66Q1VhIoJAq7s0a2Msey0eM/Jjucg6Sr9VUnTWVhV8ErXmotoiEg==

```

Utilizzo degli AWS Tools for Windows PowerShell per pubblicare in CloudWatch

Questa sezione mostra un esempio di come utilizzare i Tools for Windows PowerShell per pubblicare dati di parametri personalizzati in CloudWatch.

Questo esempio presuppone che siano state impostate le credenziali predefinite e la regione predefinita per la sessione PowerShell.

Pubblicare un parametro personalizzato nel pannello di controllo CloudWatch

Il seguente codice PowerShell inizializza un oggetto CloudWatch `MetricDatum` e lo pubblica nel servizio. Puoi visualizzare il risultato di questa operazione navigando nella [console CloudWatch](#).

```

$dat = New-Object Amazon.CloudWatch.Model.MetricDatum
$dat.Timestamp = (Get-Date).ToUniversalTime()
$dat.MetricName = "New Posts"
$dat.Unit = "Count"
$dat.Value = ".50"
Write-CWMetricData -Namespace "Usage Metrics" -MetricData $dat

```

Tenere presente quanto segue:

- Le informazioni su data e ora utilizzate per l'inizializzazione `$dat.Timestamp` devono essere nel formato Universal Time (UTC).

- Il valore che si utilizza per l'inizializzazione `$dat.Value` può essere un valore di stringa tra virgolette o un valore numerico (senza virgolette). L'esempio mostra un valore di stringa.

Vedi anche

- [Utilizzo dei servizi AWS in AWS Tools for PowerShell](#)
- [AmazonCloudWatchClient.PutMetricData](#) (documentazione di riferimento dell'SDK .NET)
- [MetricDatum](#) (Documentazione di riferimento API di servizio)
- [Amazon CloudWatch Console](#)

Uso del parametro ClientConfig nei cmdlet

Il parametro `ClientConfig` può essere usato per specificare determinate impostazioni di configurazione durante la connessione a un servizio. La maggior parte delle possibili proprietà di questo parametro è definita nella classe [Amazon.Runtime.ClientConfig](#), ereditata nelle API per i servizi AWS. Per un esempio di ereditarietà semplice, fai riferimento alla classe [Amazon.Keyspaces.AmazonKeyspacesConfig](#). Inoltre, alcuni servizi definiscono proprietà aggiuntive che sono appropriate solo per il servizio specifico. Per un esempio di definizione di proprietà aggiuntive, fai riferimento alla classe [Amazon.S3.AmazonS3Config](#) e in particolare alla proprietà `ForcePathStyle`.

Uso del parametro **ClientConfig**

Per usare il parametro `ClientConfig`, puoi specificarlo nel riga di comando come oggetto `ClientConfig` oppure puoi usare gli elenchi unitari di parametri di PowerShell per passare come unità una raccolta di valori di parametri a un comando. Questi metodi vengono mostrati negli esempi seguenti. Gli esempi presuppongono che sia stato installato e importato il modulo `AWS.Tools.S3` e che tu abbia un profilo di credenziali `[default]` con le autorizzazioni appropriate.

Definizione di un oggetto **ClientConfig**

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

Aggiunta di proprietà **ClientConfig** usando gli elenchi unitari di parametri di PowerShell

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

Get-S3Object @params
```

Uso di una proprietà non definita

Se quando usi gli elenchi unitari di parametri di PowerShell specifichi una proprietà `ClientConfig` che non esiste, AWS Tools for PowerShell non rileva l'errore fino al runtime, momento in cui restituisce un'eccezione. Modificando l'esempio precedente:

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

Get-S3Object @params
```

Questo esempio genera un'eccezione simile alla seguente:

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

Impostazione della Regione AWS

Puoi usare il parametro `ClientConfig` per impostare la Regione AWS per il comando. La regione viene impostata tramite la proprietà `RegionEndpoint`. AWS Tools for PowerShell calcola la regione da usare in base alle priorità seguenti:

1. Parametro `-Region`

2. Regione passata nel parametro `ClientConfig`
3. Stato della sessione di PowerShell
4. File `config` AWS condiviso
5. Variabili di ambiente
6. Metadati dell'istanza Amazon EC2, se abilitati.

Strumenti per esempi di PowerShell codice

Gli esempi di codice riportati in questo argomento mostrano come utilizzare AWS Tools for PowerShell with AWS.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Esempi cross-service: applicazioni di esempio che funzionano su più servizi Servizi AWS.

Esempi

- [Azioni e scenari che utilizzano Tools for PowerShell](#)

Azioni e scenari che utilizzano Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with Servizi AWS.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Servizi

- [Esempi ACM che utilizzano Tools for PowerShell](#)
- [AppStream 2.0 esempi di utilizzo di Tools for PowerShell](#)
- [Esempi di Aurora con Tools for PowerShell](#)
- [Esempi di Auto Scaling con Tools for PowerShell](#)
- [Budget AWS esempi di utilizzo di Tools for PowerShell](#)
- [AWS Cloud9 esempi di utilizzo di Tools for PowerShell](#)

- [AWS CloudFormation esempi di utilizzo di Tools for PowerShell](#)
- [CloudFront esempi di utilizzo di Tools for PowerShell](#)
- [CloudTrail esempi di utilizzo di Tools for PowerShell](#)
- [CloudWatch esempi di utilizzo di Tools for PowerShell](#)
- [CodeCommit esempi di utilizzo di Tools for PowerShell](#)
- [CodeDeploy esempi di utilizzo di Tools for PowerShell](#)
- [CodePipeline esempi di utilizzo di Tools for PowerShell](#)
- [Esempi di identità di Amazon Cognito con Tools for PowerShell](#)
- [AWS Config esempi di utilizzo di Tools for PowerShell](#)
- [Esempi di Device Farm con Tools for PowerShell](#)
- [AWS Directory Service esempi di utilizzo di Tools for PowerShell](#)
- [AWS DMS esempi di utilizzo di Tools for PowerShell](#)
- [Esempi di DynamoDB con Tools for PowerShell](#)
- [Esempi di Amazon EC2 che utilizzano Tools for PowerShell](#)
- [Esempi di Amazon ECR con Tools for PowerShell](#)
- [Esempi di Amazon ECS con Tools for PowerShell](#)
- [Esempi di Amazon EFS con Tools for PowerShell](#)
- [Esempi di Amazon EKS con Tools for PowerShell](#)
- [Elastic Load Balancing - Esempi della versione 1 che utilizzano Tools for PowerShell](#)
- [Elastic Load Balancing - Esempi della versione 2 che utilizzano Tools for PowerShell](#)
- [Esempi di Amazon FSx con Tools for PowerShell](#)
- [AWS Glue esempi di utilizzo di Tools for PowerShell](#)
- [AWS Health esempi di utilizzo di Tools for PowerShell](#)
- [Esempi IAM che utilizzano Tools for PowerShell](#)
- [Esempi di Kinesis con Tools for PowerShell](#)
- [Esempi di Lambda con Tools for PowerShell](#)
- [Esempi di Amazon ML con Tools for PowerShell](#)
- [Esempi di Macie che utilizzano Tools for PowerShell](#)
- [AWS OpsWorks esempi di utilizzo di Tools for PowerShell](#)

- [Listino prezzi AWS esempi di utilizzo di Tools for PowerShell](#)
- [Esempi di Resource Groups che utilizzano Tools for PowerShell](#)
- [Esempi di API di tagging dei Resource Groups utilizzando Tools for PowerShell](#)
- [Esempi di Route 53 che utilizzano Tools for PowerShell](#)
- [Esempi di Amazon S3 con Tools for PowerShell](#)
- [Esempi di S3 Glacier che utilizzano Tools for PowerShell](#)
- [Esempi di Amazon SES con Tools for PowerShell](#)
- [Esempi di Amazon SNS con Tools for PowerShell](#)
- [Esempi di Amazon SQS con Tools for PowerShell](#)
- [AWS STS esempi di utilizzo di Tools for PowerShell](#)
- [AWS Support esempi di utilizzo di Tools for PowerShell](#)
- [Esempi di Systems Manager che utilizzano Tools for PowerShell](#)
- [Esempi di Amazon Translate con Tools for PowerShell](#)
- [AWS WAFV2 esempi di utilizzo di Tools for PowerShell](#)
- [WorkSpaces esempi di utilizzo di Tools for PowerShell](#)

Esempi ACM che utilizzano Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with ACM.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Get-ACMCertificate

Il seguente esempio di codice mostra come utilizzare `Get-ACMCertificate`.

Strumenti per PowerShell

Esempio 1: Questo esempio mostra come restituire un certificato e la relativa catena utilizzando l'ARN del certificato.

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

- Per i dettagli sull'API, vedere [GetCertificate](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ACMCertificateDetail

Il seguente esempio di codice mostra come utilizzare `Get-ACMCertificateDetail`.

Strumenti per PowerShell

Esempio 1: restituisce i dettagli del certificato specificato.

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

Output:

```
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}
```

- Per i dettagli sull'API, vedere [DescribeCertificate](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ACMCertificateList

Il seguente esempio di codice mostra come utilizzare `Get-ACMCertificateList`

Strumenti per PowerShell

Esempio 1: recupera un elenco di tutti gli ARN dei certificati e il nome di dominio per ciascuno. Il cmdlet verrà automaticamente impaginato per recuperare tutti gli ARN. Per controllare manualmente la paginazione, utilizzare il `MaxItems` parametro - per controllare il numero di ARN di certificati restituiti per ogni chiamata di servizio e il `NextToken` parametro - per indicare il punto di partenza per ogni chiamata.

```
Get-ACMCertificateList
```

Output:

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

Esempio 2: recupera un elenco di tutti gli ARN dei certificati in cui lo stato del certificato corrisponde a quello degli stati forniti.

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

Esempio 3: Questo esempio restituisce un elenco di tutti i certificati nella regione `us-east-1` con un tipo di chiave `RSA_2048` e un utilizzo o scopo esteso della chiave di `CODE_SIGNING`. È possibile

trovare i valori per questi parametri di filtro nell'argomento di riferimento sull'API Filters: 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
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [ListCertificates](#) in AWS Tools for PowerShell Cmdlet Reference.

New-ACMCertificate

Il seguente esempio di codice mostra come utilizzare New-ACMCertificate

Strumenti per PowerShell

Esempio 1: crea un nuovo certificato. Il servizio restituisce l'ARN del nuovo certificato.

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

Output:

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

Esempio 2: crea un nuovo certificato. Il servizio restituisce l'ARN del nuovo certificato.

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

Output:

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

- Per i dettagli sull'API, vedere [RequestCertificate](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-ACMCertificate

Il seguente esempio di codice mostra come utilizzare. Remove-ACMCertificate

Strumenti per PowerShell

Esempio 1: elimina il certificato identificato dall'ARN fornito e dalla chiave privata associata. Il cmdlet richiederà la conferma prima di procedere; aggiungere l'opzione -Force per sopprimere la conferma.

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

- Per i dettagli sull'API, vedere in Cmdlet Reference. [DeleteCertificate](#) AWS Tools for PowerShell

Send-ACMValidationEmail

Il seguente esempio di codice mostra come utilizzare. Send-ACMValidationEmail

Strumenti per PowerShell

Esempio 1: richiede l'invio dell'e-mail per convalidare la proprietà del dominio per «www.example.com». Se il valore \$ della shell ConfirmPreference è impostato su 'Medium' o inferiore, il cmdlet richiederà una conferma prima di procedere. Aggiungi l'interruttore -Force per sopprimere le richieste di conferma.

```
$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
```

- Per i dettagli sull'API, vedere [ResendValidationEmail](#) in Cmdlet Reference.AWS Tools for PowerShell

AppStream 2.0 esempi di utilizzo di Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with AppStream 2.0.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Add-APSResourceTag

Il seguente esempio di codice mostra come utilizzare Add-APSResourceTag.

Strumenti per PowerShell

Esempio 1: questo esempio aggiunge un tag di risorsa alla AppStream risorsa

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

Output:

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

- Per i dettagli sull'API, vedere [TagResource](#) in AWS Tools for PowerShell Cmdlet Reference.

Copy-APSIImage

Il seguente esempio di codice mostra come utilizzare. Copy-APSIImage

Strumenti per PowerShell

Esempio 1: questo esempio copia un'immagine in un'altra regione

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

Output:

```
TestImageCopy
```

- Per i dettagli sull'API, vedere [CopyImage](#) in AWS Tools for PowerShell Cmdlet Reference.

Disable-APSUser

Il seguente esempio di codice mostra come utilizzare. Disable-APSUser

Strumenti per PowerShell

Esempio 1: questo esempio disabilita un utente in USERPOOL

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

- Per i dettagli sull'API, vedere [DisableUser](#) in AWS Tools for PowerShell Cmdlet Reference.

Enable-APUser

Il seguente esempio di codice mostra come utilizzare. Enable-APUser

Strumenti per PowerShell

Esempio 1: Questo esempio abilita un utente disabile in USERPOOL

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

- Per i dettagli sull'API, vedere [EnableUser](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-APSAssociatedFleetList

Il seguente esempio di codice mostra come utilizzare. Get-APSAssociatedFleetList

Strumenti per PowerShell

Esempio 1: questo esempio mostra la flotta associata a uno stack

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

Output:

```
PowershellFleet
```

- Per i dettagli sull'API, vedere [ListAssociatedFleets](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-APSAssociatedStackList

Il seguente esempio di codice mostra come utilizzare. Get-APSAssociatedStackList

Strumenti per PowerShell

Esempio 1: questo esempio mostra lo stack associato a una flotta

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

Output:

```
PowershellStack
```

- Per i dettagli sull'API, vedere [ListAssociatedStacks](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-APSDirectoryConfigList

Il seguente esempio di codice mostra come utilizzare. Get-APSDirectoryConfigList

Strumenti per PowerShell

Esempio 1: questo esempio visualizza le configurazioni di directory create in AppStream

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

Output:

```
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
```

- Per i dettagli sull'API, vedere [DescribeDirectoryConfigs](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-APSFleetList

Il seguente esempio di codice mostra come utilizzare. Get-APSFleetList

Strumenti per PowerShell

Esempio 1: questo esempio mostra i dettagli di una flotta

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

Output:

```
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

```

- Per i dettagli sull'API, vedere [DescribeFleets](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-APSIImageBuilderList

Il seguente esempio di codice mostra come utilizzare `Get-APSIImageBuilderList`

Strumenti per PowerShell

Esempio 1: questo esempio visualizza i dettagli di un ImageBuilder

```
Get-APSIImageBuilderList -Name TestImage
```

Output:

```

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

```

- Per i dettagli sull'API, vedere [DescribeImageBuilders](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-APSIImageList

Il seguente esempio di codice mostra come utilizzare. `Get-APSIImageList`

Strumenti per PowerShell

Esempio 1: questo esempio mostra AppStream immagini private

```

Get-APSIImageList -Type PRIVATE | select DisplayName, ImageBuilderName, Visibility,
arn

```

Output:

```

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

```

- Per i dettagli sull'API, vedere [DescribeImages](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-APSIImagePermission

Il seguente esempio di codice mostra come utilizzare. `Get-APSIImagePermission`

Strumenti per PowerShell

Esempio 1: questo esempio visualizza le autorizzazioni relative alle immagini su un'immagine condivisa AppStream

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

Output:

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

- Per i dettagli sull'API, vedere [DescribeImagePermissions](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-APSSessionList

Il seguente esempio di codice mostra come utilizzare `Get-APSSessionList`

Strumenti per PowerShell

Esempio 1: questo esempio mostra l'elenco delle sessioni per una flotta

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

Output:

```
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
```

- Per i dettagli sull'API, vedere [DescribeSessions](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-APSSStackList

Il seguente esempio di codice mostra come utilizzare. Get-APSSStackList

Strumenti per PowerShell

Esempio 1: questo esempio mostra un elenco di AppStream Stack

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

Output:

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

- Per i dettagli sull'API, vedere [DescribeStacks](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-APSTagsForResourceList

Il seguente esempio di codice mostra come utilizzare. Get-APSTagsForResourceList

Strumenti per PowerShell

Esempio 1: questo esempio visualizza i tag su una AppStream risorsa

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

Output:

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

- Per i dettagli sull'API, vedere [ListTagsForResource](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-APSUsageReportSubscription

Il seguente esempio di codice mostra come utilizzare. Get-APSUsageReportSubscription

Strumenti per PowerShell

Esempio 1: questo esempio visualizza i dettagli AppStreamUsageReport di configurazione

```
Get-APSUsageReportSubscription
```

Output:

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

- Per i dettagli sull'API, vedere [DescribeUsageReportSubscriptions](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-APSUser

Il seguente esempio di codice mostra come utilizzare. Get-APSUser

Strumenti per PowerShell

Esempio 1: questo esempio visualizza un elenco di utenti con stato abilitato

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

Output:

```
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

- Per i dettagli sull'API, vedere [DescribeUsers](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-APStackAssociation

Il seguente esempio di codice mostra come utilizzare `Get-APStackAssociation`

Strumenti per PowerShell

Esempio 1: questo esempio visualizza l'elenco degli utenti assegnati a uno stack

```
Get-APStackAssociation -StackName PowershellStack
```

Output:

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

- Per i dettagli sull'API, vedere [DescribeUserStackAssociations](#) in AWS Tools for PowerShell Cmdlet Reference.

New-APSDirectoryConfig

Il seguente esempio di codice mostra come utilizzare `New-APSDirectoryConfig`

Strumenti per PowerShell

Esempio 1: questo esempio crea una configurazione di directory in AppStream

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

Output:


```

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

```

- Per i dettagli sull'API, vedere [CreateDirectoryConfigin](#) AWS Tools for PowerShell Cmdlet Reference.

New-APSFleet

Il seguente esempio di codice mostra come utilizzare. `New-APSFleet`

Strumenti per PowerShell

Esempio 1: questo esempio crea un nuovo AppStream parco veicoli

```

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

```

Output:

```

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

```

- Per i dettagli sull'API, vedere [CreateFleet](#) in AWS Tools for PowerShell Cmdlet Reference.

New-APSIImageBuilder

Il seguente esempio di codice mostra come utilizzare `New-APSIImageBuilder`

Strumenti per PowerShell

Esempio 1: Questo esempio crea un Image Builder in AppStream

```

New-APSIImageBuilder -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

```

Output:

```

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

```



```

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}
  
```

- Per i dettagli sull'API, vedere [CreateStack](#) in AWS Tools for PowerShell Cmdlet Reference.

New-APSSstreamingURL

Il seguente esempio di codice mostra come utilizzare. `New-APSSstreamingURL`

Strumenti per PowerShell

Esempio 1: questo esempio crea un URL di streaming di Stack

```

New-APSSstreamingURL -StackName SessionScriptTest -FleetName SessionScriptNew -UserId
TestUser
  
```

Output:

```

Expires          StreamingURL
-----
12/27/2019 12:43:37 PM https://appstream2.us-east-1.aws.amazon.com/authenticate?
parameters=eyJ0eXB1IjoiRU5EX1VTRVIiLCJleHBpcmVzIjoiMTU3NzQ1MDYxNyIsImF3c0FjY291bnRJZCI6IjM5M...
```

- Per i dettagli sull'API, vedere [CreateStreamingURL](#) in AWS Tools for PowerShell Cmdlet Reference.

New-APSUsageReportSubscription

Il seguente esempio di codice mostra come utilizzare. `New-APSUsageReportSubscription`

Strumenti per PowerShell

Esempio 1: questo esempio abilita i report di AppStream utilizzo

```
New-APSUsageReportSubscription
```

Output:

```
S3BucketName          Schedule
-----
appstream-logs-us-east-1-123456789012-sik2hnxe DAILY
```

- Per i dettagli sull'API, vedere [CreateUsageReportSubscription](#) in AWS Tools for PowerShell Cmdlet Reference.

New-APSUser

Il seguente esempio di codice mostra come utilizzare. New-APSUser

Strumenti per PowerShell

Esempio 1: questo esempio crea un utente in USERPOOL

```
New-APSUser -UserName Test@lab.com -AuthenticationType USERPOOL -FirstName 'kt' -
LastName 'aws' -Select ^UserName
```

Output:

```
Test@lab.com
```

- Per i dettagli sull'API, vedere [CreateUser](#) in AWS Tools for PowerShell Cmdlet Reference.

Register-APSFleet

Il seguente esempio di codice mostra come utilizzare. Register-APSFleet

Strumenti per PowerShell

Esempio 1: questo esempio registra la flotta con uno stack

```
Register-APSFleet -StackName TestStack -FleetName TestFleet -Region us-west-2
```

- Per i dettagli sull'API, vedere [AssociateFleet](#) in AWS Tools for PowerShell Cmdlet Reference.

Register-APSUserStackBatch

Il seguente esempio di codice mostra come utilizzare Register-APSUserStackBatch

Strumenti per PowerShell

Esempio 1: questo esempio assegna lo stack a un utente in USERPOOL

```
Register-APSUserStackBatch -UserStackAssociation  
@{AuthenticationType="USERPOOL";SendEmailNotification=  
$False;StackName="PowershellStack";UserName="TestUser1@lab.com"}
```

- Per i dettagli sull'API, vedere [BatchAssociateUserStack](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-APSDirectoryConfig

Il seguente esempio di codice mostra come utilizzare Remove-APSDirectoryConfig

Strumenti per PowerShell

Esempio 1: questo esempio rimuove la configurazione della AppStream directory

```
Remove-APSDirectoryConfig -DirectoryName contoso.com
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DeleteDirectoryConfig](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-APSFleet

Il seguente esempio di codice mostra come utilizzare. Remove-APSFleet

Strumenti per PowerShell

Esempio 1: questo esempio rimuove le eliminazioni da un parco AppStream

```
Remove-APSFleet -Name TestFleet -Region us-west-2
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DeleteFleet](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-APSIImage

Il seguente esempio di codice mostra come utilizzare. Remove-APSIImage

Strumenti per PowerShell

Esempio 1: questo esempio elimina un'immagine

```
Remove-APSIImage -Name TestImage -Region us-west-2
```

Output:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSIImage (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

```

- Per i dettagli sull'API, vedere [DeleteImage](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-APSIImageBuilder

Il seguente esempio di codice mostra come utilizzare `Remove-APSIImageBuilder`

Strumenti per PowerShell

Esempio 1: questo esempio elimina un ImageBuilder

```
Remove-APSIImageBuilder -Name TestIB -Region us-west-2
```

Output:

```

Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSIImageBuilder (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
```

- Per i dettagli sull'API, vedere [DeleteImageBuilder](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-APSIImagePermission

Il seguente esempio di codice mostra come utilizzare. `Remove-APSIImagePermission`

Strumenti per PowerShell

Esempio 1: questo esempio rimuove le autorizzazioni di un'immagine

```
Remove-APSIImagePermission -Name Powershell -SharedAccountId 123456789012
```

Output:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSIImagePermission (DeleteImagePermissions)" on
target "Powershell".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): A
```

- Per i dettagli sull'API, vedere [DeleteImagePermissions](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-APSResourceTag

Il seguente esempio di codice mostra come utilizzare. `Remove-APSResourceTag`

Strumenti per PowerShell

Esempio 1: questo esempio rimuove un tag di risorsa dalla AppStream risorsa

```
Remove-APSRResourceTag -ResourceArn arn:aws:appstream:us-east-1:123456789012:stack/SessionScriptTest -TagKey StackState
```

Output:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSRResourceTag (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
```

- Per i dettagli sull'API, vedere [UntagResource](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-APSSStack

Il seguente esempio di codice mostra come utilizzare. Remove-APSSStack

Strumenti per PowerShell

Esempio 1: questo esempio elimina uno stack

```
Remove-APSSStack -Name TestStack -Region us-west-2
```

Output:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSSStack (DeleteStack)" on target "TestStack".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): A
```

- Per i dettagli sull'API, vedere [DeleteStack](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-APSUsageReportSubscription

Il seguente esempio di codice mostra come utilizzare. Remove-APSUsageReportSubscription

Strumenti per PowerShell

Esempio 1: questo esempio disattiva l'abbonamento AppStream all'Usage Report

```
Remove-APSUsageReportSubscription
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DeleteUsageReportSubscription](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-APSUser

Il seguente esempio di codice mostra come utilizzare. Remove-APSUser

Strumenti per PowerShell

Esempio 1: questo esempio elimina un utente da USERPOOL

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

Output:

```
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
```

- Per i dettagli sull'API, vedere [DeleteUser](#) in AWS Tools for PowerShell Cmdlet Reference.

Revoke-APSSession

Il seguente esempio di codice mostra come utilizzare. Revoke-APSSession

Strumenti per PowerShell

Esempio 1: questo esempio revoca una sessione a fleet AppStream

```
Revoke-APSSession -SessionId 6cd2f9a3-f948-4aa1-8014-8a7dcde14877
```

- Per i dettagli sull'API, vedere [ExpireSession](#) in AWS Tools for PowerShell Cmdlet Reference.

Start-APSFleet

Il seguente esempio di codice mostra come utilizzare. Start-APSFleet

Strumenti per PowerShell

Esempio 1: questo esempio avvia una flotta

```
Start-APSFleet -Name PowershellFleet
```

- Per i dettagli sull'API, vedere [StartFleet](#) in AWS Tools for PowerShell Cmdlet Reference.

Start-APSIImageBuilder

Il seguente esempio di codice mostra come utilizzare. Start-APSIImageBuilder

Strumenti per PowerShell

Esempio 1: questo esempio avvia un ImageBuilder

```
Start-APSIImageBuilder -Name TestImage
```

Output:

```
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

```

- Per i dettagli sull'API, vedere [StartImageBuilder](#) in AWS Tools for PowerShell Cmdlet Reference.

Stop-APSFleet

Il seguente esempio di codice mostra come utilizzare. Stop-APSFleet

Strumenti per PowerShell

Esempio 1: questo esempio arresta una flotta

```
Stop-APSFleet -Name PowershellFleet
```

- Per i dettagli sull'API, vedere [StopFleet](#) in AWS Tools for PowerShell Cmdlet Reference.

Stop-APSIImageBuilder

Il seguente esempio di codice mostra come utilizzare. Stop-APSIImageBuilder

Strumenti per PowerShell

Esempio 1: questo esempio interrompe un ImageBuilder

```
Stop-APSIImageBuilder -Name TestImage
```

Output:

```

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

```

- Per i dettagli sull'API, vedere [StopImageBuilder](#) in AWS Tools for PowerShell Cmdlet Reference.

Unregister-APSFleet

Il seguente esempio di codice mostra come utilizzare `Unregister-APSFleet`

Strumenti per PowerShell

Esempio 1: questo esempio annulla la registrazione di un parco veicoli dallo stack

```
Unregister-APSFleet -StackName TestStack -FleetName TestFleet -Region us-west-2
```

- Per i dettagli sull'API, vedere [DisassociateFleet](#) in AWS Tools for PowerShell Cmdlet Reference.

Unregister-APSUserStackBatch

Il seguente esempio di codice mostra come utilizzare `Unregister-APSUserStackBatch`

Strumenti per PowerShell

Esempio 1: questo esempio rimuove un utente da uno stack assegnato

```
Unregister-APSUserStackBatch -UserStackAssociation
@{AuthenticationType="USERPOOL";SendEmailNotification=
$False;StackName="PowershellStack";UserName="TestUser1@lab.com"}
```

- Per i dettagli sull'API, vedere [BatchDisassociateUserStack](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-APSDirectoryConfig

Il seguente esempio di codice mostra come utilizzare. Update-APSDirectoryConfig

Strumenti per PowerShell

Esempio 1: questo esempio aggiorna la configurazione della directory creata in AppStream

```
Update-APSDirectoryConfig -ServiceAccountCredentials_AccountName contoso
\ServiceAccount -ServiceAccountCredentials_AccountPassword MyPass@1$@#
-DirectoryName contoso.com -OrganizationalUnitDistinguishedName
"OU=AppStreamNew,OU=Contoso,DC=Contoso,DC=com"
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [UpdateDirectoryConfig](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-APSFleet

Il seguente esempio di codice mostra come utilizzare. Update-APSFleet

Strumenti per PowerShell

Esempio 1: questo esempio aggiorna le proprietà di un parco veicoli

```
Update-APSFleet -Name PowershellFleet -EnableDefaultInternetAccess $True -
DisconnectTimeoutInSecond 950
```

Output:

```

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

```

- Per i dettagli sull'API, vedere [UpdateFleet](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-APSIImagePermission

Il seguente esempio di codice mostra come utilizzare. Update-APSIImagePermission

Strumenti per PowerShell

Esempio 1: questo esempio condivide un' AppStream immagine con un altro account

```

Update-APSIImagePermission -Name Powershell -SharedAccountId 123456789012 -
ImagePermissions_AllowFleet $True -ImagePermissions_AllowImageBuilder $True

```

- Per i dettagli sull'API, vedere [UpdateImagePermissions](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-APSSStack

Il seguente esempio di codice mostra come utilizzare. Update-APSSStack

Strumenti per PowerShell

Esempio 1: questo esempio aggiorna (abilita) la persistenza delle impostazioni dell'applicazione e le cartelle Home in uno stack

```
Update-APSSStack -Name PowershellStack -ApplicationSettings_Enabled $True
-ApplicationSettings_SettingsGroup PowershellStack -StorageConnector
@{ConnectorType="HOMEFOLDERS"}
```

Output:

```
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}
```

- Per i dettagli sull'API, vedere [UpdateStack](#) in AWS Tools for PowerShell Cmdlet Reference.

Esempi di Aurora con Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell con Aurora.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Get-RDSOrderableDBInstanceOption

Il seguente esempio di codice mostra come utilizzare `Get-RDSOrderableDBInstanceOption`.

Strumenti per PowerShell

Esempio 1: Questo esempio elenca le versioni del motore DB che supportano una classe di istanza DB specifica in un Regione AWS.

```
$params = @{
    Engine = 'aurora-postgresql'
    DBInstanceClass = 'db.r5.large'
    Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

Esempio 2: Questo esempio elenca le classi di istanze DB supportate per una versione specifica del motore DB in un Regione AWS.

```
$params = @{
    Engine = 'aurora-postgresql'
    EngineVersion = '13.6'
    Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

- Per i dettagli sull'API, vedere [DescribeOrderableDB InstanceOptions](#) in AWS Tools for PowerShell Cmdlet Reference.

Esempi di Auto Scaling con Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando l' AWS Tools for PowerShell Auto Scaling.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Add-ASLoadBalancer

Il seguente esempio di codice mostra come utilizzare `Add-ASLoadBalancer`.

Strumenti per PowerShell

Esempio 1: questo esempio collega il load balancer specificato al gruppo Auto Scaling specificato.

```
Add-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- Per i dettagli sull'API, vedere [AttachLoadBalancers](#) in AWS Tools for PowerShell Cmdlet Reference.

Complete-ASLifecycleAction

Il seguente esempio di codice mostra come utilizzare `Complete-ASLifecycleAction`.

Strumenti per PowerShell

Esempio 1: questo esempio completa l'azione del ciclo di vita specificata.

```
Complete-ASLifecycleAction -LifecycleHookName myLifecycleHook -  
AutoScalingGroupName my-asg -LifecycleActionResult CONTINUE -LifecycleActionToken  
bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- Per i dettagli sull'API, vedere [CompleteLifecycleAction](#) in AWS Tools for PowerShell Cmdlet Reference.

Disable-ASMetricsCollection

Il seguente esempio di codice mostra come utilizzare `Disable-ASMetricsCollection`

Strumenti per PowerShell

Esempio 1: questo esempio disabilita il monitoraggio delle metriche specificate per il gruppo Auto Scaling specificato.

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg -Metric @("GroupMinSize",  
"GroupMaxSize")
```

Esempio 2: questo esempio disabilita il monitoraggio di tutte le metriche per il gruppo Auto Scaling specificato.

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg
```

- Per i dettagli sull'API, vedere [DisableMetricsCollection](#) in AWS Tools for PowerShell Cmdlet Reference.

Dismount-ASInstance

Il seguente esempio di codice mostra come utilizzare `Dismount-ASInstance`

Strumenti per PowerShell

Esempio 1: Questo esempio scollega l'istanza specificata dal gruppo Auto Scaling specificato e riduce la capacità desiderata in modo che Auto Scaling non avvii un'istanza sostitutiva.

```
Dismount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg -  
ShouldDecrementDesiredCapacity $true
```

Output:

```

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        :

```

Esempio 2: Questo esempio scollega l'istanza specificata dal gruppo Auto Scaling specificato senza ridurre la capacità desiderata. Auto Scaling avvia un'istanza sostitutiva.

```

Dismount-ASInstance -InstanceId i-7bf746a2 -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false

```

Output:

```

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        :

```

- Per i dettagli sull'API, vedere [DetachInstances](#) in AWS Tools for PowerShell Cmdlet Reference.

Dismount-ASLoadBalancer

Il seguente esempio di codice mostra come utilizzare `Dismount-ASLoadBalancer`

Strumenti per PowerShell

Esempio 1: Questo esempio scollega il sistema di bilanciamento del carico specificato dal gruppo Auto Scaling specificato.

```
Dismount-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- Per i dettagli sull'API, vedere [DetachLoadBalancers](#) in AWS Tools for PowerShell Cmdlet Reference.

Enable-ASMetricsCollection

Il seguente esempio di codice mostra come utilizzare. Enable-ASMetricsCollection

Strumenti per PowerShell

Esempio 1: questo esempio abilita il monitoraggio delle metriche specificate per il gruppo Auto Scaling specificato.

```
Enable-ASMetricsCollection -Metric @("GroupMinSize", "GroupMaxSize") -  
AutoScalingGroupName my-asg -Granularity 1Minute
```

Esempio 2: questo esempio abilita il monitoraggio di tutte le metriche per il gruppo Auto Scaling specificato.

```
Enable-ASMetricsCollection -AutoScalingGroupName my-asg -Granularity 1Minute
```

- Per i dettagli sull'API, vedere [EnableMetricsCollection](#) in AWS Tools for PowerShell Cmdlet Reference.

Enter-ASStandby

Il seguente esempio di codice mostra come utilizzare. Enter-ASStandby

Strumenti per PowerShell

Esempio 1: Questo esempio mette l'istanza specificata in modalità standby e riduce la capacità desiderata in modo che Auto Scaling non avvii un'istanza sostitutiva.

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -  
ShouldDecrementDesiredCapacity $true
```

Output:

```
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        :
```

Esempio 2: questo esempio mette l'istanza specificata in modalità standby senza ridurre la capacità desiderata. Auto Scaling avvia un'istanza sostitutiva.

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -  
ShouldDecrementDesiredCapacity $false
```

Output:

```
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        :
```

- Per i dettagli sull'API, vedere [EnterStandby](#) in AWS Tools for PowerShell Cmdlet Reference.

Exit-ASStandby

Il seguente esempio di codice mostra come utilizzare. Exit-ASStandby

Strumenti per PowerShell

Esempio 1: Questo esempio sposta l'istanza specificata dalla modalità standby.

```
Exit-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

Output:

```
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        :
```

- Per i dettagli sull'API, vedere [ExitStandby](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ASAccountLimit

Il seguente esempio di codice mostra come utilizzare. Get-ASAccountLimit

Strumenti per PowerShell

Esempio 1: questo esempio descrive i limiti delle risorse Auto Scaling per il tuo AWS account.

```
Get-ASAccountLimit
```

Output:

```
MaxNumberOfAutoScalingGroups : 20
```



```
MaxNumberOfLaunchConfigurations : 100
```

- Per i dettagli sull'API, vedere [DescribeAccountLimits](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ASAdjustmentType

Il seguente esempio di codice mostra come utilizzare. `Get-ASAdjustmentType`

Strumenti per PowerShell

Esempio 1: Questo esempio descrive i tipi di regolazione supportati da Auto Scaling.

```
Get-ASAdjustmentType
```

Output:

```
Type
----
ChangeInCapacity
ExactCapacity
PercentChangeInCapacity
```

- Per i dettagli sull'API, vedere [DescribeAdjustmentTypes](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ASAutoScalingGroup

Il seguente esempio di codice mostra come utilizzare. `Get-ASAutoScalingGroup`

Strumenti per PowerShell

Esempio 1: questo esempio elenca i nomi dei gruppi di Auto Scaling.

```
Get-ASAutoScalingGroup | format-table -property AutoScalingGroupName
```

Output:

```
AutoScalingGroupName
-----
```

```
my-asg-1
my-asg-2
my-asg-3
my-asg-4
my-asg-5
my-asg-6
```

Esempio 2: Questo esempio descrive il gruppo Auto Scaling specificato.

```
Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1
```

Output:

```
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
```

Esempio 3: Questo esempio descrive i due gruppi di Auto Scaling specificati.

```
Get-ASAutoScalingGroup -AutoScalingGroupName @"my-asg-1", "my-asg-2"
```

Esempio 4: Questo esempio descrive le istanze Auto Scaling per il gruppo Auto Scaling specificato.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1).Instances
```

Esempio 5: questo esempio descrive tutti i gruppi di Auto Scaling.

```
Get-ASAutoScalingGroup
```

Esempio 6: Questo esempio descrive tutti i gruppi di Auto Scaling, in batch da 10.

```
$nextToken = $null
do {
  Get-ASAutoScalingGroup -NextToken $nextToken -MaxRecord 10
  $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

Esempio 7: Questo LaunchTemplate esempio descrive il gruppo Auto Scaling specificato. Questo esempio presuppone che l'opzione «Instance Purchase options» sia impostata su «Aderisci al modello di lancio». Nel caso in cui questa opzione sia impostata su «Combina opzioni di acquisto e tipi di istanze», è LaunchTemplate possibile accedervi utilizzando ". MixedInstancesPolicy LaunchTemplate«proprietà.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-ag-1).LaunchTemplate
```

Output:

LaunchTemplateId	LaunchTemplateName	Version
lt-06095fd619cb40371	test-launch-template	\$Default

- Per i dettagli sull'API, vedere [DescribeAutoScalingGroups](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ASAutoScalingInstance

Il seguente esempio di codice mostra come utilizzare. Get-ASAutoScalingInstance

Strumenti per PowerShell

Esempio 1: questo esempio elenca gli ID delle istanze di Auto Scaling.

```
Get-ASAutoScalingInstance | format-table -property InstanceId
```

Output:

```
InstanceId
-----
i-12345678
i-87654321
i-abcd1234
```

Esempio 2: questo esempio descrive l'istanza Auto Scaling specificata.

```
Get-ASAutoScalingInstance -InstanceId i-12345678
```

Output:

```
AutoScalingGroupName    : my-asg
AvailabilityZone         : us-west-2b
HealthStatus            : HEALTHY
InstanceId               : i-12345678
LaunchConfigurationName : my-lc
LifecycleState          : InService
```

Esempio 3: Questo esempio descrive le due istanze di Auto Scaling specificate.

```
Get-ASAutoScalingInstance -InstanceId @"(i-12345678", "i-87654321")
```

Esempio 4: Questo esempio descrive le istanze Auto Scaling per il gruppo Auto Scaling specificato.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg).Instances | Get-ASAutoScalingInstance
```

Esempio 5: questo esempio descrive tutte le istanze di Auto Scaling.

```
Get-ASAutoScalingInstance
```

Esempio 6: questo esempio descrive tutte le istanze di Auto Scaling, in batch da 10.

```
$nextToken = $null
do {
  Get-ASAutoScalingInstance -NextToken $nextToken -MaxRecord 10
  $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Per i dettagli sull'API, vedere [DescribeAutoScalingInstances](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ASAutoScalingNotificationType

Il seguente esempio di codice mostra come utilizzare. Get-ASAutoScalingNotificationType

Strumenti per PowerShell

Esempio 1: Questo esempio elenca i tipi di notifica supportati da Auto Scaling.

```
Get-ASAutoScalingNotificationType
```

Output:

```
autoscaling:EC2_INSTANCE_LAUNCH
autoscaling:EC2_INSTANCE_LAUNCH_ERROR
autoscaling:EC2_INSTANCE_TERMINATE
autoscaling:EC2_INSTANCE_TERMINATE_ERROR
autoscaling:TEST_NOTIFICATION
```

- Per i dettagli sull'API, vedere [DescribeAutoScalingNotificationTypes](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ASLaunchConfiguration

Il seguente esempio di codice mostra come utilizzare. Get-ASLaunchConfiguration

Strumenti per PowerShell

Esempio 1: questo esempio elenca i nomi delle configurazioni di avvio.

```
Get-ASLaunchConfiguration | format-table -property LaunchConfigurationName
```

Output:

```

LaunchConfigurationName
-----
my-lc-1
my-lc-2
my-lc-3
my-lc-4
my-lc-5

```

Esempio 2: Questo esempio descrive la configurazione di avvio specificata.

```
Get-ASLaunchConfiguration -LaunchConfigurationName my-lc-1
```

Output:

```

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                      :

```

Esempio 3: Questo esempio descrive le due configurazioni di avvio specificate.

```
Get-ASLaunchConfiguration -LaunchConfigurationName @("my-lc-1", "my-lc-2")
```

Esempio 4: Questo esempio descrive tutte le configurazioni di avvio.

```
Get-ASLaunchConfiguration
```

Esempio 5: questo esempio descrive tutte le configurazioni di lancio, in batch da 10.

```
$nextToken = $null
do {
    Get-ASLaunchConfiguration -NextToken $nextToken -MaxRecord 10
    $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Per i dettagli sull'API, vedere [DescribeLaunchConfigurations](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ASLifecycleHook

Il seguente esempio di codice mostra come utilizzare. Get-ASLifecycleHook

Strumenti per PowerShell

Esempio 1: questo esempio descrive l'hook del ciclo di vita specificato.

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName myLifecycleHook
```

Output:

```
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
```

Esempio 2: Questo esempio descrive tutti gli hook del ciclo di vita per il gruppo Auto Scaling specificato.

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg
```

Esempio 3: Questo esempio descrive tutti gli hook del ciclo di vita per tutti i gruppi di Auto Scaling.

```
Get-ASLifecycleHook
```

- Per i dettagli sull'API, vedere [DescribeLifecycleHooks](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-ASLifecycleHookType

Il seguente esempio di codice mostra come utilizzare. Get-ASLifecycleHookType

Strumenti per PowerShell

Esempio 1: questo esempio elenca i tipi di hook del ciclo di vita supportati da Auto Scaling.

```
Get-ASLifecycleHookType
```

Output:

```
autoscaling:EC2_INSTANCE_LAUNCHING  
auto-scaling:EC2_INSTANCE_TERMINATING
```

- Per i dettagli sull'API, vedere [DescribeLifecycleHookTypes](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ASLoadBalancer

Il seguente esempio di codice mostra come utilizzare. Get-ASLoadBalancer

Strumenti per PowerShell

Esempio 1: Questo esempio descrive i sistemi di bilanciamento del carico per il gruppo Auto Scaling specificato.

```
Get-ASLoadBalancer -AutoScalingGroupName my-asg
```

Output:


```
LoadBalancerName    State
-----
my-lb                Added
```

- Per i dettagli sull'API, vedere [DescribeLoadBalancers](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ASMetricCollectionType

Il seguente esempio di codice mostra come utilizzare `Get-ASMetricCollectionType`

Strumenti per PowerShell

Esempio 1: questo esempio elenca i tipi di raccolta di metriche supportati da Auto Scaling.

```
(Get-ASMetricCollectionType).Metrics
```

Output:

```
Metric
-----
GroupMinSize
GroupMaxSize
GroupDesiredCapacity
GroupInServiceInstances
GroupPendingInstances
GroupTerminatingInstances
GroupStandbyInstances
GroupTotalInstances
```

Esempio 2: questo esempio elenca le granularità corrispondenti.

```
(Get-ASMetricCollectionType).Granularities
```

Output:

```
Granularity
-----
1Minute
```

- Per i dettagli sull'API, vedere [DescribeMetricCollectionTypes](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ASNotificationConfiguration

Il seguente esempio di codice mostra come utilizzare `Get-ASNotificationConfiguration`

Strumenti per PowerShell

Esempio 1: Questo esempio descrive le azioni di notifica associate al gruppo Auto Scaling specificato.

```
Get-ASNotificationConfiguration -AutoScalingGroupName my-asg | format-list
```

Output:

```
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
```

Esempio 2: Questo esempio descrive le azioni di notifica associate a tutti i gruppi di Auto Scaling.

```
Get-ASNotificationConfiguration
```

- Per i dettagli sull'API, vedere [DescribeNotificationConfigurations](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ASPolicy

Il seguente esempio di codice mostra come utilizzare `Get-ASPolicy`

Strumenti per PowerShell

Esempio 1: Questo esempio descrive tutte le politiche per il gruppo Auto Scaling specificato.

```
Get-ASPolicy -AutoScalingGroupName my-asg
```

Output:

```
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     : {}
```

Esempio 2: Questo esempio descrive le politiche specificate per il gruppo Auto Scaling specificato.

```
Get-ASPolicy -AutoScalingGroupName my-asg -PolicyName @("myScaleOutPolicy",
"myScaleInPolicy")
```

Esempio 3: Questo esempio descrive tutte le politiche per tutti i gruppi di Auto Scaling.

```
Get-ASPolicy
```

- Per i dettagli sull'API, vedere [DescribePolicies](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ASScalingActivity

Il seguente esempio di codice mostra come utilizzare `Get-ASScalingActivity`

Strumenti per PowerShell

Esempio 1: Questo esempio descrive le attività di scalabilità delle ultime sei settimane per il gruppo Auto Scaling specificato.

```
Get-ASScalingActivity -AutoScalingGroupName my-asg
```

Output:

```

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        :

```

Esempio 2: questo esempio descrive l'attività di scalabilità specificata.

```
Get-ASScalingActivity -ActivityId "063308ae-aa22-4a9b-94f4-9fae4EXAMPLE"
```

Esempio 3: Questo esempio descrive le attività di scaling delle ultime sei settimane per tutti i tuoi gruppi di Auto Scaling.

```
Get-ASScalingActivity
```

- Per i dettagli sull'API, vedere [DescribeScalingActivities](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ASScalingProcessType

Il seguente esempio di codice mostra come utilizzare. `Get-ASScalingProcessType`

Strumenti per PowerShell

Esempio 1: Questo esempio elenca i tipi di processo supportati da Auto Scaling.

```
Get-ASScalingProcessType
```

Output:

```
ProcessName
-----
AZRebalance
AddToLoadBalancer
AlarmNotification
HealthCheck
Launch
ReplaceUnhealthy
ScheduledActions
Terminate
```

- Per i dettagli sull'API, vedere [DescribeScalingProcessTypes](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ASScheduledAction

Il seguente esempio di codice mostra come utilizzare. `Get-ASScheduledAction`

Strumenti per PowerShell

Esempio 1: Questo esempio descrive le azioni di ridimensionamento pianificate per il gruppo Auto Scaling specificato.

```
Get-ASScheduledAction -AutoScalingGroupName my-asg
```

Output:

```
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
```

Esempio 2: questo esempio descrive le azioni di ridimensionamento pianificate specificate.

```
Get-ASScheduledAction -ScheduledActionName @("myScheduledScaleOut",
"myScheduledScaleIn")
```

Esempio 3: questo esempio descrive le azioni di ridimensionamento pianificate che iniziano entro l'ora specificata.

```
Get-ASScheduledAction -StartTime "2015-12-01T08:00:00Z"
```

Esempio 4: questo esempio descrive le azioni di ridimensionamento pianificate che terminano entro l'ora specificata.

```
Get-ASScheduledAction -EndTime "2015-12-30T08:00:00Z"
```

Esempio 5: Questo esempio descrive le azioni di ridimensionamento pianificate per tutti i gruppi di Auto Scaling.

```
Get-ASScheduledAction
```

- Per i dettagli sull'API, vedere [DescribeScheduledActions](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ASTag

Il seguente esempio di codice mostra come utilizzare. Get-ASTag

Strumenti per PowerShell

Esempio 1: Questo esempio descrive i tag con un valore chiave «myTag» o «myTag2». I valori possibili per il nome del filtro sono ", auto-scaling-group 'key', 'value' e ". propagate-at-launch La sintassi utilizzata da questo esempio richiede la PowerShell versione 3 o successiva.

```
Get-ASTag -Filter @( @{ Name="key"; Values=@("myTag", "myTag2") } )
```

Output:

```
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
```

Esempio 2: con la PowerShell versione 2, è necessario utilizzare New-Object per creare il filtro per il parametro Filter.

```
$keys = New-Object string[] 2
$keys[0] = "myTag"
$keys[1] = "myTag2"
$filter = New-Object Amazon.AutoScaling.Model.Filter
$filter.Name = "key"
$filter.Values = $keys
Get-ASTag -Filter @( $filter )
```

Esempio 3: Questo esempio descrive tutti i tag per tutti i gruppi di Auto Scaling.

```
Get-ASTag
```

- Per i dettagli sull'API, vedere [DescribeTags](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ASTerminationPolicyType

Il seguente esempio di codice mostra come utilizzare. Get-ASTerminationPolicyType

Strumenti per PowerShell

Esempio 1: Questo esempio elenca le politiche di terminazione supportate da Auto Scaling.

```
Get-ASTerminationPolicyType
```

Output:

```
ClosestToNextInstanceHour
Default
NewestInstance
OldestInstance
OldestLaunchConfiguration
```

- Per i dettagli sull'API, vedere [DescribeTerminationPolicyTypes](#) in AWS Tools for PowerShell Cmdlet Reference.

Mount-ASInstance

Il seguente esempio di codice mostra come utilizzare. Mount-ASInstance

Strumenti per PowerShell

Esempio 1: questo esempio collega l'istanza specificata al gruppo Auto Scaling specificato. Auto Scaling aumenta automaticamente la capacità desiderata del gruppo Auto Scaling.

```
Mount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

- Per i dettagli sull'API, vedere [AttachInstances](#) in AWS Tools for PowerShell Cmdlet Reference.

New-ASAutoScalingGroup

Il seguente esempio di codice mostra come utilizzare. New-ASAutoScalingGroup

Strumenti per PowerShell

Esempio 1: Questo esempio crea un gruppo Auto Scaling con il nome e gli attributi specificati. La capacità predefinita desiderata è la dimensione minima. Pertanto, questo gruppo Auto Scaling avvia due istanze, una in ciascuna delle due zone di disponibilità specificate.

```
New-ASAutoScalingGroup -AutoScalingGroupName my-asg -LaunchConfigurationName my-lc -MinSize 2 -MaxSize 6 -AvailabilityZone @("us-west-2a", "us-west-2b")
```

- Per i dettagli sull'API, vedere [CreateAutoScalingGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

New-ASLaunchConfiguration

Il seguente esempio di codice mostra come utilizzare. New-ASLaunchConfiguration

Strumenti per PowerShell

Esempio 1: Questo esempio crea una configurazione di avvio denominata 'my-lc'. Le istanze EC2 lanciate dai gruppi di Auto Scaling che utilizzano questa configurazione di avvio utilizzano il tipo di istanza, l'AMI, il gruppo di sicurezza e il ruolo IAM specificati.

```
New-ASLaunchConfiguration -LaunchConfigurationName my-lc -InstanceType "m3.medium" -ImageId "ami-12345678" -SecurityGroup "sg-12345678" -IamInstanceProfile "myIamRole"
```

- Per i dettagli sull'API, consulta Cmdlet [CreateLaunchConfiguration](#) Reference AWS Tools for PowerShell .

Remove-ASAutoScalingGroup

Il seguente esempio di codice mostra come utilizzare. Remove-ASAutoScalingGroup

Strumenti per PowerShell

Esempio 1: Questo esempio elimina il gruppo Auto Scaling specificato se non ha istanze in esecuzione. Prima di procedere con l'operazione, viene richiesta una conferma.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg
```

Output:

```
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"):
```

Esempio 2: se si specifica il parametro Force, non viene richiesta la conferma prima che l'operazione proceda.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -Force
```

Esempio 3: Questo esempio elimina il gruppo Auto Scaling specificato e termina tutte le istanze in esecuzione in esso contenute.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -ForceDelete $true -Force
```

- Per i dettagli sull'API, vedere [DeleteAutoScalingGroup](#) in Cmdlet Reference.AWS Tools for PowerShell

Remove-ASLaunchConfiguration

Il seguente esempio di codice mostra come utilizzare Remove-ASLaunchConfiguration

Strumenti per PowerShell

Esempio 1: Questo esempio elimina la configurazione di avvio specificata se non è collegata a un gruppo Auto Scaling. Prima di procedere con l'operazione, viene richiesta una conferma.

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc
```

Output:

```
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"):
```

Esempio 2: se si specifica il parametro `Force`, non viene richiesta la conferma prima che l'operazione proceda.

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc -Force
```

- Per i dettagli sull'API, vedere [DeleteLaunchConfiguration](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-ASLifecycleHook

Il seguente esempio di codice mostra come utilizzare `Remove-ASLifecycleHook`

Strumenti per PowerShell

Esempio 1: Questo esempio elimina l'hook del ciclo di vita specificato per il gruppo Auto Scaling specificato. Prima di procedere con l'operazione, viene richiesta una conferma.

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName myLifecycleHook
```

Output:

```
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"):
```

Esempio 2: se si specifica il parametro `Force`, non viene richiesta la conferma prima che l'operazione proceda.

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName myLifecycleHook -Force
```

- Per i dettagli sull'API, vedere [DeleteLifecycleHook](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-ASNotificationConfiguration

Il seguente esempio di codice mostra come utilizzare. Remove-ASNotificationConfiguration

Strumenti per PowerShell

Esempio 1: questo esempio elimina l'azione di notifica specificata. Prima di procedere con l'operazione, viene richiesta una conferma.

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN
"arn:aws:sns:us-west-2:123456789012:my-topic"
```

Output:

```
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"):
```

Esempio 2: se si specifica il parametro Force, non viene richiesta la conferma prima che l'operazione proceda.

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN
"arn:aws:sns:us-west-2:123456789012:my-topic" -Force
```

- Per i dettagli sull'API, vedere [DeleteNotificationConfiguration](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-ASPolicy

Il seguente esempio di codice mostra come utilizzare. Remove-ASPolicy

Strumenti per PowerShell

Esempio 1: Questo esempio elimina la politica specificata per il gruppo Auto Scaling specificato. Prima di procedere con l'operazione, viene richiesta una conferma.

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy
```

Output:

```
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"):
```

Esempio 2: se si specifica il parametro Force, non viene richiesta la conferma prima che l'operazione proceda.

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy -Force
```

- Per i dettagli sull'API, vedere [DeletePolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-ASScheduledAction

Il seguente esempio di codice mostra come utilizzare Remove-ASScheduledAction

Strumenti per PowerShell

Esempio 1: Questo esempio elimina l'azione pianificata specificata per il gruppo Auto Scaling specificato. Prima di procedere con l'operazione, viene richiesta una conferma.

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction
"myScheduledAction"
```

Output:

```
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"):
```

Esempio 2: se si specifica il parametro Force, non viene richiesta la conferma prima che l'operazione proceda.

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction  
"myScheduledAction" -Force
```

- Per i dettagli sull'API, vedere [DeleteScheduledAction](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-ASTag

Il seguente esempio di codice mostra come utilizzare. Remove-ASTag

Strumenti per PowerShell

Esempio 1: Questo esempio rimuove il tag specificato dal gruppo Auto Scaling specificato. Prima di procedere con l'operazione, viene richiesta una conferma. La sintassi utilizzata da questo esempio richiede la PowerShell versione 3 o successiva.

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";  
Key="myTag" } )
```

Output:

```
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"):
```

Esempio 2: se si specifica il parametro Force, non viene richiesta la conferma prima che l'operazione proceda.

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";  
Key="myTag" } ) -Force
```

Esempio 3: con Powershell versione 2, è necessario utilizzare New-Object per creare il tag per il parametro Tag.

```
$tag = New-Object Amazon.AutoScaling.Model.Tag  
$tag.ResourceType = "auto-scaling-group"  
$tag.ResourceId = "my-asg"
```

```
$tag.Key = "myTag"  
Remove-ASTag -Tag $tag -Force
```

- Per i dettagli sull'API, vedere [DeleteTags](#) in AWS Tools for PowerShell Cmdlet Reference.

Resume-ASProcess

Il seguente esempio di codice mostra come utilizzare. Resume-ASProcess

Strumenti per PowerShell

Esempio 1: Questo esempio riprende il processo di Auto Scaling specificato per il gruppo Auto Scaling specificato.

```
Resume-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"
```

Esempio 2: Questo esempio riprende tutti i processi di Auto Scaling sospesi per il gruppo Auto Scaling specificato.

```
Resume-ASProcess -AutoScalingGroupName my-asg
```

- Per i dettagli sull'API, vedere [ResumeProcesses](#) in Cmdlet Reference.AWS Tools for PowerShell

Set-ASDesiredCapacity

Il seguente esempio di codice mostra come utilizzare. Set-ASDesiredCapacity

Strumenti per PowerShell

Esempio 1: Questo esempio imposta la dimensione del gruppo Auto Scaling specificato.

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2
```

Esempio 2: Questo esempio imposta la dimensione del gruppo Auto Scaling specificato e attende il completamento del periodo di cooldown prima di passare alla nuova dimensione.

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2 -HonorCooldown  
$true
```

- Per i dettagli sull'API, vedere [SetDesiredCapacity](#) in Cmdlet Reference.AWS Tools for PowerShell

Set-ASInstanceHealth

Il seguente esempio di codice mostra come utilizzare. Set-ASInstanceHealth

Strumenti per PowerShell

Esempio 1: questo esempio imposta lo stato dell'istanza specificata su «Non sana», mettendola fuori servizio. Auto Scaling termina e sostituisce l'istanza.

```
Set-ASInstanceHealth -HealthStatus Unhealthy -InstanceId i-93633f9b
```

Esempio 2: Questo esempio imposta lo stato dell'istanza specificata su «Healthy», mantenendola in servizio. Qualsiasi periodo di tolleranza per il controllo dello stato di salute per il gruppo Auto Scaling non viene rispettato.

```
Set-ASInstanceHealth -HealthStatus Healthy -InstanceId i-93633f9b -  
ShouldRespectGracePeriod $false
```

- Per i dettagli sull'API, vedere [SetInstanceHealth](#) in AWS Tools for PowerShell Cmdlet Reference.

Set-ASInstanceProtection

Il seguente esempio di codice mostra come utilizzare. Set-ASInstanceProtection

Strumenti per PowerShell

Esempio 1: questo esempio abilita la protezione dell'istanza per l'istanza specificata.

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -  
ProtectedFromScaleIn $true
```

Esempio 2: questo esempio disattiva la protezione dell'istanza per l'istanza specificata.

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -  
ProtectedFromScaleIn $false
```


- Per i dettagli sull'API, vedere [SetInstanceProtection](#) in AWS Tools for PowerShell Cmdlet Reference.

Set-ASTag

Il seguente esempio di codice mostra come utilizzare. Set-ASTag

Strumenti per PowerShell

Esempio 1: Questo esempio aggiunge un singolo tag al gruppo Auto Scaling specificato. La chiave del tag è 'myTag' e il valore del tag è ". myTagValue Auto Scaling propaga questo tag alle istanze EC2 successive lanciate dal gruppo Auto Scaling. La sintassi utilizzata da questo esempio richiede la versione 3 o successiva. PowerShell

```
Set-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";  
Key="myTag"; Value="myTagValue"; PropagateAtLaunch=$true} )
```

Esempio 2: con PowerShell la versione 2, è necessario utilizzare New-Object per creare il tag per il parametro Tag.

```
$tag = New-Object Amazon.AutoScaling.Model.Tag  
$tag.ResourceType = "auto-scaling-group"  
$tag.ResourceId = "my-asg"  
$tag.Key = "myTag"  
$tag.Value = "myTagValue"  
$tag.PropagateAtLaunch = $true  
Set-ASTag -Tag $tag
```

- Per i dettagli sull'API, vedere [CreateOrUpdateTags](#) in AWS Tools for PowerShell Cmdlet Reference.

Start-ASPolicy

Il seguente esempio di codice mostra come utilizzare. Start-ASPolicy

Strumenti per PowerShell

Esempio 1: questo esempio esegue la politica specificata per il gruppo Auto Scaling specificato.

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy"
```

Esempio 2: Questo esempio esegue il criterio specificato per il gruppo Auto Scaling specificato, dopo aver atteso il completamento del periodo di cooldown.

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy" -
HonorCooldown $true
```

- Per i dettagli sull'API, vedere [ExecutePolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Stop-ASInstanceInAutoScalingGroup

Il seguente esempio di codice mostra come utilizzare `Stop-ASInstanceInAutoScalingGroup`

Strumenti per PowerShell

Esempio 1: Questo esempio termina l'istanza specificata e riduce la capacità desiderata del relativo gruppo Auto Scaling in modo che Auto Scaling non avvii un'istanza sostitutiva.

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $true
```

Output:

```
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        :
```

Esempio 2: Questo esempio termina l'istanza specificata senza ridurre la capacità desiderata del relativo gruppo Auto Scaling. Auto Scaling avvia un'istanza sostitutiva.

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $false
```

Output:

```
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       :
```

- Per i dettagli sull'API, vedere [TerminateInstanceInAutoScalingGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

Suspend-ASProcess

Il seguente esempio di codice mostra come utilizzare. Suspend-ASProcess

Strumenti per PowerShell

Esempio 1: Questo esempio sospende il processo di Auto Scaling specificato per il gruppo Auto Scaling specificato.

```
Suspend-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"
```

Esempio 2: Questo esempio sospende tutti i processi di Auto Scaling per il gruppo Auto Scaling specificato.

```
Suspend-ASProcess -AutoScalingGroupName my-asg
```

- Per i dettagli sull'API, vedere [SuspendProcesses](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-ASAutoScalingGroup

Il seguente esempio di codice mostra come utilizzare. Update-ASAutoScalingGroup

Strumenti per PowerShell

Esempio 1: Questo esempio aggiorna la dimensione minima e massima del gruppo Auto Scaling specificato.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -MaxSize 5 -MinSize 1
```

Esempio 2: questo esempio aggiorna il periodo di cooldown predefinito del gruppo Auto Scaling specificato.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -DefaultCooldown 10
```

Esempio 3: questo esempio aggiorna le zone di disponibilità del gruppo Auto Scaling specificato.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -AvailabilityZone @("us-west-2a", "us-west-2b")
```

Esempio 4: questo esempio aggiorna il gruppo Auto Scaling specificato per utilizzare i controlli di integrità Elastic Load Balancing.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -HealthCheckType ELB -  
HealthCheckGracePeriod 60
```

- Per i dettagli sull'API, vedere [UpdateAutoScalingGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

Write-ASLifecycleActionHeartbeat

Il seguente esempio di codice mostra come utilizzare. Write-ASLifecycleActionHeartbeat

Strumenti per PowerShell

Esempio 1: questo esempio registra un battito cardiaco per l'azione del ciclo di vita specificata. In questo modo l'istanza rimane in sospeso fino al completamento dell'azione personalizzata.

```
Write-ASLifecycleActionHeartbeat -AutoScalingGroupName my-asg -LifecycleHookName  
myLifecycleHook -LifecycleActionToken bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- Per i dettagli sull'API, vedere [RecordLifecycleActionHeartbeat](#) in AWS Tools for PowerShell Cmdlet Reference.

Write-ASLifecycleHook

Il seguente esempio di codice mostra come utilizzare Write-ASLifecycleHook

Strumenti per PowerShell

Esempio 1: questo esempio aggiunge l'hook del ciclo di vita specificato al gruppo Auto Scaling specificato.

```
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"
```

- Per i dettagli sull'API, vedere [PutLifecycleHook](#) in AWS Tools for PowerShell Cmdlet Reference.

Write-ASNotificationConfiguration

Il seguente esempio di codice mostra come utilizzare Write-ASNotificationConfiguration

Strumenti per PowerShell

Esempio 1: questo esempio configura il gruppo Auto Scaling specificato per inviare una notifica all'argomento SNS specificato quando avvia istanze EC2.

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType  
"autoscaling:EC2_INSTANCE_LAUNCH" -TopicARN "arn:aws:sns:us-west-2:123456789012:my-  
topic"
```

Esempio 2: questo esempio configura il gruppo Auto Scaling specificato per inviare una notifica all'argomento SNS specificato quando avvia o termina le istanze EC2.

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType
@("autoscaling:EC2_INSTANCE_LAUNCH", "autoscaling:EC2_INSTANCE_TERMINATE") -
TopicARN "arn:aws:sns:us-west-2:123456789012:my-topic"
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [PutNotificationConfiguration](#) AWS Tools for PowerShell

Write-ASScalingPolicy

Il seguente esempio di codice mostra come utilizzare. `Write-ASScalingPolicy`

Strumenti per PowerShell

Esempio 1: Questo esempio aggiunge il criterio specificato al gruppo Auto Scaling specificato. Il tipo di regolazione specificato determina come interpretare il `ScalingAdjustment` parametro. Con 'ChangeInCapacity', un valore positivo aumenta la capacità del numero specificato di istanze e un valore negativo diminuisce la capacità del numero di istanze specificato.

```
Write-ASScalingPolicy -AutoScalingGroupName my-asg -AdjustmentType
"ChangeInCapacity" -PolicyName "myScaleInPolicy" -ScalingAdjustment -1
```

Output:

```
arn:aws:autoscaling:us-west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-
e1d769fc24ef:autoScalingGroupName/my-asg
:policyName/myScaleInPolicy
```

- Per i dettagli sull'API, vedere [PutScalingPolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Write-ASScheduledUpdateGroupAction

Il seguente esempio di codice mostra come utilizzare. `Write-ASScheduledUpdateGroupAction`

Strumenti per PowerShell

Esempio 1: Questo esempio crea o aggiorna un'azione pianificata una tantum per modificare la capacità desiderata all'ora di inizio specificata.

```
Write-ASScheduledUpdateGroupAction -AutoScalingGroupName my-asg -ScheduledActionName
"myScheduledAction" -StartTime "2015-12-01T00:00:00Z" -DesiredCapacity 10
```

- Per i dettagli sull'API, vedere [PutScheduledUpdateGroupAction](#) in AWS Tools for PowerShell Cmdlet Reference.

Budget AWS esempi di utilizzo di Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with Budget AWS.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

New-BGTBudget

Il seguente esempio di codice mostra come utilizzare `New-BGTBudget`.

Strumenti per PowerShell

Esempio 1: crea un nuovo budget con i vincoli di budget e di tempo specificati con notifiche e-mail.

```
$notification = @{
    NotificationType = "ACTUAL"
    ComparisonOperator = "GREATER_THAN"
    Threshold = 80
}

$addressObject = @{
    Address = @"user@domain.com"
```

```
SubscriptionType = "EMAIL"
}

$subscriber = New-Object Amazon.Budgets.Model.NotificationWithSubscribers
$subscriber.Notification = $notification
$subscriber.Subscribers.Add($addressObject)

$startDate = [datetime]::new(2017,09,25)
$endDate = [datetime]::new(2017,10,25)

New-BGTBudget -Budget_BudgetName "Tester" -Budget_BudgetType COST -
CostTypes_IncludeTax $true -Budget_TimeUnit MONTHLY -BudgetLimit_Unit USD -
TimePeriod_Start $startDate -TimePeriod_End $endDate -AccountId 123456789012 -
BudgetLimit_Amount 200 -NotificationsWithSubscriber $subscriber
```

- Per i dettagli sull'API, vedere [CreateBudget](#) in AWS Tools for PowerShell Cmdlet Reference.

AWS Cloud9 esempi di utilizzo di Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with AWS Cloud9.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Get-C9EnvironmentData

Il seguente esempio di codice mostra come utilizzare `Get-C9EnvironmentData`.

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene informazioni sugli ambienti di sviluppo AWS Cloud9 specificati.

```
Get-C9EnvironmentData -EnvironmentId
685f892f431b45c2b28cb69eadcdb0EX,1980b80e5f584920801c09086667f0EX
```

Output:

```
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
```

Esempio 2: Questo esempio ottiene informazioni sullo stato del ciclo di vita dell'ambiente di sviluppo Cloud9 specificato AWS .

```
(Get-C9EnvironmentData -EnvironmentId 685f892f431b45c2b28cb69eadcdb0EX).Lifecycle
```

Output:

```
FailureResource Reason Status
-----
                          -----
                          CREATED
```

- Per i dettagli sull'API, vedere [DescribeEnvironments](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-C9EnvironmentList

Il seguente esempio di codice mostra come utilizzare. Get-C9EnvironmentList

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene un elenco di identificatori dell'ambiente di AWS sviluppo Cloud9 disponibili.

```
Get-C9EnvironmentList
```

Output:

```
685f892f431b45c2b28cb69eadcdb0EX  
1980b80e5f584920801c09086667f0EX
```

- Per i dettagli sull'API, vedere [ListEnvironments](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-C9EnvironmentMembershipList

Il seguente esempio di codice mostra come utilizzare. Get-C9EnvironmentMembershipList

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene informazioni sui membri dell'ambiente di sviluppo AWS Cloud9 specificato.

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeeaaa8a04bfde8cEX
```

Output:

```
EnvironmentId : ffd88420d4824eeeeaaa8a04bfde8cEX  
LastAccess    : 1/1/0001 12:00:00 AM  
Permissions   : read-write  
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser  
UserId        : AIDAJ3BA602FMJWCXHEX  
  
EnvironmentId : ffd88420d4824eeeeaaa8a04bfde8cEX  
LastAccess    : 1/1/0001 12:00:00 AM
```

```
Permissions    : owner
UserArn        : arn:aws:iam::123456789012:user/MyDemoUser
UserId         : AIDAJ3LOROMOUCTBSUGEX
```

Esempio 2: Questo esempio ottiene informazioni sul proprietario dell'ambiente di sviluppo AWS Cloud9 specificato.

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeeaea8a04bfde8cEX -
Permission owner
```

Output:

```
EnvironmentId : ffd88420d4824eeeeaea8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions    : owner
UserArn        : arn:aws:iam::123456789012:user/MyDemoUser
UserId         : AIDAJ3LOROMOUCTBSUGEX
```

Esempio 3: Questo esempio ottiene informazioni sul membro dell'ambiente specificato per più ambienti di sviluppo AWS Cloud9.

```
Get-C9EnvironmentMembershipList -UserArn arn:aws:iam::123456789012:user/MyDemoUser
```

Output:

```
EnvironmentId : ffd88420d4824eeeeaea8a04bfde8cEX
LastAccess    : 1/17/2018 7:48:14 PM
Permissions    : owner
UserArn        : arn:aws:iam::123456789012:user/MyDemoUser
UserId         : AIDAJ3LOROMOUCTBSUGEX

EnvironmentId : 1980b80e5f584920801c09086667f0EX
LastAccess    : 1/16/2018 11:21:24 PM
Permissions    : owner
UserArn        : arn:aws:iam::123456789012:user/MyDemoUser
UserId         : AIDAJ3LOROMOUCTBSUGEX
```

- Per i dettagli sull'API, vedere [DescribeEnvironmentMemberships](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-C9EnvironmentStatus

Il seguente esempio di codice mostra come utilizzare. `Get-C9EnvironmentStatus`

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene informazioni sullo stato per l'ambiente di sviluppo AWS Cloud9 specificato.

```
Get-C9EnvironmentStatus -EnvironmentId 349c86d4579e4e7298d500ff57a6b2EX
```

Output:

```
Message                Status
-----                -
Environment is ready to use ready
```

- Per i dettagli sull'API, vedere [DescribeEnvironmentStatus](#) in AWS Tools for PowerShell Cmdlet Reference.

New-C9EnvironmentEC2

Il seguente esempio di codice mostra come utilizzare. `New-C9EnvironmentEC2`

Strumenti per PowerShell

Esempio 1: questo esempio crea un AWS ambiente di sviluppo Cloud9 con le impostazioni specificate, avvia un'istanza Amazon Elastic Compute Cloud (Amazon EC2) e quindi si connette dall'istanza all'ambiente.

```
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
```

Output:

```
ffd88420d4824eeeeaaa8a04bfde8cEX
```

- [Per i dettagli sull'API, consulta CreateEnvironmentEc 2 in Cmdlet Reference.AWS Tools for PowerShell](#)

New-C9EnvironmentMembership

Il seguente esempio di codice mostra come utilizzare. `New-C9EnvironmentMembership`

Strumenti per PowerShell

Esempio 1: Questo esempio aggiunge il membro dell'ambiente specificato all'ambiente di sviluppo AWS Cloud9 specificato.

```
New-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/AnotherDemoUser  
-EnvironmentId ffd88420d4824eeeeaeaa8a04bfde8cEX -Permission read-write
```

Output:

```
EnvironmentId : ffd88420d4824eeeeaeaa8a04bfde8cEX  
LastAccess    : 1/1/0001 12:00:00 AM  
Permissions   : read-write  
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser  
UserId        : AIDAJ3BA602FMJWCXHEX
```

- Per i dettagli sull'API, vedere [CreateEnvironmentMembership](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-C9Environment

Il seguente esempio di codice mostra come utilizzare. `Remove-C9Environment`

Strumenti per PowerShell

Esempio 1: questo esempio elimina l'ambiente di sviluppo AWS Cloud9 specificato. Se un'istanza Amazon EC2 è connessa all'ambiente, interrompe anche l'istanza.

```
Remove-C9Environment -EnvironmentId ffd88420d4824eeeeaeaa8a04bfde8cEX
```

- Per i dettagli sull'API, consulta AWS Tools for PowerShell Cmdlet [DeleteEnvironment](#) Reference.

Remove-C9EnvironmentMembership

Il seguente esempio di codice mostra come utilizzare. `Remove-C9EnvironmentMembership`

Strumenti per PowerShell

Esempio 1: Questo esempio elimina il membro dell'ambiente specificato dall'ambiente di sviluppo AWS Cloud9 specificato.

```
Remove-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeeaeaa8a04bfde8cEX
```

- Per i dettagli sull'API, vedere [DeleteEnvironmentMembership](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-C9Environment

Il seguente esempio di codice mostra come utilizzare. Update-C9Environment

Strumenti per PowerShell

Esempio 1: Questo esempio modifica le impostazioni specificate dell'ambiente di sviluppo AWS Cloud9 esistente specificato.

```
Update-C9Environment -EnvironmentId ffd88420d4824eeeeaeaa8a04bfde8cEX -Description
"My changed demonstration development environment." -Name my-changed-demo-env
```

- Per i dettagli sull'API, vedere [UpdateEnvironment](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-C9EnvironmentMembership

Il seguente esempio di codice mostra come utilizzare. Update-C9EnvironmentMembership

Strumenti per PowerShell

Esempio 1: Questo esempio modifica le impostazioni del membro di ambiente esistente specificato per l'ambiente di sviluppo AWS Cloud9 specificato.

```
Update-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeeaeaa8a04bfde8cEX -Permission read-
only
```

Output:

```
EnvironmentId : ffd88420d4824eeeeaea8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-only
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA602FMJWCWXHEX
```

- Per i dettagli sull'API, vedere [UpdateEnvironmentMembership](#) in AWS Tools for PowerShell Cmdlet Reference.

AWS CloudFormation esempi di utilizzo di Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with AWS CloudFormation.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Get-CFNStack

Il seguente esempio di codice mostra come utilizzare `Get-CFNStack`.

Strumenti per PowerShell

Esempio 1: restituisce una raccolta di istanze Stack che descrivono tutti gli stack dell'utente.

```
Get-CFNStack
```

Esempio 2: restituisce un'istanza Stack che descrive lo stack specificato

```
Get-CFNStack -StackName "myStack"
```

Esempio 3: restituisce una raccolta di istanze Stack che descrivono tutti gli stack dell'utente utilizzando il paging manuale. Il token iniziale per la pagina successiva viene recuperato dopo ogni chiamata con \$null che indica che non rimangono altri dettagli da recuperare.

```
$nextToken = $null
do {
    Get-CFNStack -NextToken $nextToken
    $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Per i dettagli sull'API, vedere [DescribeStacks](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-CFNStackEvent

Il seguente esempio di codice mostra come utilizzare. Get-CFNStackEvent

Strumenti per PowerShell

Esempio 1: restituisce tutti gli eventi relativi allo stack per lo stack specificato.

```
Get-CFNStackEvent -StackName "myStack"
```

Esempio 2: restituisce tutti gli eventi relativi allo stack per lo stack specificato utilizzando il paging manuale a partire dal token specificato. Il token iniziale per la pagina successiva viene recuperato dopo ogni chiamata con \$null che indica che non rimangono altri eventi da recuperare.

```
$nextToken = $null
do {
    Get-CFNStack -StackName "myStack" -NextToken $nextToken
    $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Per i dettagli sull'API, vedere [DescribeStackEvents](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-CFNStackResource

Il seguente esempio di codice mostra come utilizzare. Get-CFNStackResource

Strumenti per PowerShell

Esempio 1: restituisce la descrizione di una risorsa identificata nel modello associato allo stack specificato dall'ID logico «myDBInstance».

```
Get-CFNStackResource -StackName "myStack" -LogicalResourceId "MyDBInstance"
```

- Per i dettagli sull'API, vedere [DescribeStackResource](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CFNStackResourceList

Il seguente esempio di codice mostra come utilizzare. Get-CFNStackResourceList

Strumenti per PowerShell

Esempio 1: restituisce le descrizioni AWS delle risorse per un massimo di 100 risorse associate allo stack specificato. Per ottenere i dettagli di tutte le risorse associate a uno stack, usa Get-CFNStackResourceSummary, che supporta anche la paginazione manuale dei risultati.

```
Get-CFNStackResourceList -StackName "myStack"
```

Esempio 2: restituisce la descrizione dell'istanza Amazon EC2 identificata nel modello associato allo stack specificato dall'ID logico «Ec2Instance».

```
Get-CFNStackResourceList -StackName "myStack" -LogicalResourceId "Ec2Instance"
```

Esempio 3: restituisce la descrizione di un massimo di 100 risorse associate allo stack contenente un'istanza Amazon EC2 identificata dall'ID di istanza «i-123456». Per ottenere i dettagli di tutte le risorse associate a uno stack, usa Get-CFN, che supporta anche la paginazione manuale StackResourceSummary dei risultati.

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456"
```

Esempio 4: restituisce la descrizione dell'istanza Amazon EC2 identificata dall'ID logico «Ec2Instance» nel modello per uno stack. Lo stack viene identificato utilizzando l'ID della risorsa fisica di una risorsa che contiene, in questo caso anche un'istanza Amazon EC2 con ID di istanza «i-123456». È inoltre possibile utilizzare una risorsa fisica diversa per identificare lo stack in base al contenuto del modello, ad esempio un bucket Amazon S3.

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456" -LogicalResourceId  
"Ec2Instance"
```

- Per i dettagli sull'API, vedere [DescribeStackResources](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CFNStackResourceSummary

Il seguente esempio di codice mostra come utilizzare. `Get-CFNStackResourceSummary`

Strumenti per PowerShell

Esempio 1: restituisce le descrizioni di tutte le risorse associate allo stack specificato.

```
Get-CFNStackResourceSummary -StackName "myStack"
```

Esempio 2: restituisce le descrizioni di tutte le risorse associate allo stack specificato utilizzando la paginazione manuale dei risultati. Il token iniziale per la pagina successiva viene recuperato dopo ogni chiamata con `$null` che indica che non rimangono altri dettagli da recuperare.

```
$nextToken = $null  
do {  
    Get-CFNStackResourceSummary -StackName "myStack" -NextToken $nextToken  
    $nextToken = $AWSHistory.LastServiceResponse.NextToken  
} while ($nextToken -ne $null)
```

- Per i dettagli sull'API, vedere [ListStackResources](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-CFNStackSummary

Il seguente esempio di codice mostra come utilizzare. `Get-CFNStackSummary`

Strumenti per PowerShell

Esempio 1: restituisce informazioni di riepilogo per tutti gli stack.

```
Get-CFNStackSummary
```

Esempio 2: restituisce informazioni di riepilogo per tutti gli stack attualmente in fase di creazione.

```
Get-CFNStackSummary -StackStatusFilter "CREATE_IN_PROGRESS"
```

Esempio 3: restituisce informazioni di riepilogo per tutti gli stack attualmente in fase di creazione o aggiornamento.

```
Get-CFNStackSummary -StackStatusFilter @("CREATE_IN_PROGRESS", "UPDATE_IN_PROGRESS")
```

Esempio 4: restituisce informazioni di riepilogo per tutti gli stack attualmente in fase di creazione o aggiornamento utilizzando l'impaginazione manuale dei risultati. Il token iniziale per la pagina successiva viene recuperato dopo ogni chiamata con \$null che indica che non rimangono altri dettagli da recuperare.

```
$nextToken = $null
do {
    Get-CFNStackSummary -StackStatusFilter @("CREATE_IN_PROGRESS",
    "UPDATE_IN_PROGRESS") -NextToken $nextToken
    $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Per i dettagli sull'API, vedere [ListStacks](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-CFNTemplate

Il seguente esempio di codice mostra come utilizzare. Get-CFNTemplate

Strumenti per PowerShell

Esempio 1: restituisce il modello associato allo stack specificato.

```
Get-CFNTemplate -StackName "myStack"
```

- Per i dettagli sull'API, vedere [GetTemplate](#) in AWS Tools for PowerShell Cmdlet Reference.

Measure-CFNTemplateCost

Il seguente esempio di codice mostra come utilizzare. Measure-CFNTemplateCost

Strumenti per PowerShell

Esempio 1: restituisce l'URL di un calcolatore mensile AWS semplice con una stringa di query che descrive le risorse necessarie per eseguire il modello. Il modello è ottenuto dall'URL Amazon S3 specificato e dal singolo parametro di personalizzazione applicato. Il parametro può anche essere specificato utilizzando «Key» e «Value» anziché «» e ParameterKey «». ParameterValue

```
Measure-CFNTemplateCost -TemplateURL https://s3.amazonaws.com/mytemplates/
templatefile.template `
    -Region us-west-1 `
    -Parameter @{ ParameterKey="KeyName";
ParameterValue="myKeyPairName" }
```

Esempio 2: restituisce l'URL di un calcolatore mensile AWS semplice con una stringa di query che descrive le risorse necessarie per eseguire il modello. Il modello viene analizzato in base al contenuto fornito e ai parametri di personalizzazione applicati (l'esempio presuppone che il contenuto del modello abbia dichiarato due parametri, " e 'KeyName'). InstanceType I parametri di personalizzazione possono anche essere specificati utilizzando 'Key' e 'Value' anziché " e 'ParameterKey'. ParameterValue

```
Measure-CFNTemplateCost -TemplateBody "{TEMPLATE CONTENT HERE}" `
    -Parameter @( @{ ParameterKey="KeyName";
ParameterValue="myKeyPairName" }, `
                @{ ParameterKey="InstanceType";
ParameterValue="m1.large" })
```

Esempio 3: utilizza New-Object per creare il set di parametri del modello e restituisce un URL di AWS Simple Monthly Calculator con una stringa di query che descrive le risorse necessarie per eseguire il modello. Il modello viene analizzato a partire dal contenuto fornito, con parametri di personalizzazione (questo esempio presuppone che il contenuto del modello abbia dichiarato due parametri, " e "). 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 )
```

- Per i dettagli sull'API, vedere [EstimateTemplateCost](#) in AWS Tools for PowerShell Cmdlet Reference.

New-CFNStack

Il seguente esempio di codice mostra come utilizzare New-CFNStack

Strumenti per PowerShell

Esempio 1: crea un nuovo stack con il nome specificato. Il modello viene analizzato a partire dal contenuto fornito con parametri di personalizzazione («PK1» e «PK2» rappresentano i nomi dei parametri dichiarati nel contenuto del modello, «PV1» e «PV2» rappresentano i valori di tali parametri. I parametri di personalizzazione possono essere specificati anche utilizzando «Key» e «Value» anziché «» e «». ParameterKey ParameterValue Se la creazione dello stack fallisce, non verrà ripristinato.

```
New-CFNStack -StackName "myStack" `
    -TemplateBody "{TEMPLATE CONTENT HERE}" `
    -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
    @{ ParameterKey="PK2"; ParameterValue="PV2" } ) `
    -DisableRollback $true
```

Esempio 2: crea un nuovo stack con il nome specificato. Il modello viene analizzato a partire dal contenuto fornito con parametri di personalizzazione («PK1» e «PK2» rappresentano i nomi dei parametri dichiarati nel contenuto del modello, «PV1» e «PV2» rappresentano i valori di tali parametri. I parametri di personalizzazione possono essere specificati anche utilizzando «Key» e «Value» anziché «» e «». ParameterKey ParameterValue Se la creazione dello stack fallisce, verrà ripristinato.

```
$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"
```

Esempio 3: crea un nuovo stack con il nome specificato. Il modello è ottenuto dall'URL di Amazon S3 con parametri di personalizzazione («PK1» rappresenta il nome di un parametro dichiarato nel contenuto del modello, «PV1» rappresenta il valore del parametro. I parametri di personalizzazione possono anche essere specificati utilizzando «Key» e «Value» anziché «» e «». ParameterKey ParameterValue Se la creazione dello stack fallisce, verrà ripristinato (come specificare - DisableRollback \$false).

```
New-CFNStack -StackName "myStack" `
             -TemplateURL https://s3.amazonaws.com/mytemplates/templatefile.template
             `
             -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Esempio 4: crea un nuovo stack con il nome specificato. Il modello è ottenuto dall'URL di Amazon S3 con parametri di personalizzazione («PK1» rappresenta il nome di un parametro dichiarato nel contenuto del modello, «PV1» rappresenta il valore del parametro. I parametri di personalizzazione possono anche essere specificati utilizzando «Key» e «Value» anziché «» e «». ParameterKey ParameterValue Se la creazione dello stack fallisce, verrà ripristinato (come specificare - DisableRollback \$false). La notifica specificata AENs riceverà gli eventi pubblicati relativi allo stack.

```
New-CFNStack -StackName "myStack" `
             -TemplateURL https://s3.amazonaws.com/mytemplates/templatefile.template
             `
             -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" } `
             -NotificationARN @( "arn1", "arn2" )
```

- Per i dettagli sull'API, vedere [CreateStack](#) in Cmdlet Reference.AWS Tools for PowerShell

Remove-CFNStack

Il seguente esempio di codice mostra come utilizzare. Remove-CFNStack

Strumenti per PowerShell

Esempio 1: elimina lo stack specificato.

```
Remove-CFNStack -StackName "myStack"
```

- Per i dettagli sull'API, vedere [DeleteStack](#) in AWS Tools for PowerShell Cmdlet Reference.

Resume-CFNUpdateRollback

Il seguente esempio di codice mostra come utilizzare. Resume-CFNUpdateRollback

Strumenti per PowerShell

Esempio 1: continua il rollback dello stack denominato, che dovrebbe trovarsi nello stato 'UPDATE_ROLLBACK_FAILED'. Se il rollback continuato ha esito positivo, lo stack entrerà nello stato 'UPDATE_ROLLBACK_COMPLETE'.

```
Resume-CFNUpdateRollback -StackName "myStack"
```

- Per i dettagli [ContinueUpdateRollback](#) sull'AWS Tools for PowerShell API, vedere in Cmdlet Reference.

Stop-CFNUpdateStack

Il seguente esempio di codice mostra come utilizzare. Stop-CFNUpdateStack

Strumenti per PowerShell

Esempio 1: annulla un aggiornamento sullo stack specificato.

```
Stop-CFNUpdateStack -StackName "myStack"
```

- Per i dettagli sull'API, vedere [CancelUpdateStack](#) in AWS Tools for PowerShell Cmdlet Reference.

Test-CFNTemplate

Il seguente esempio di codice mostra come utilizzare. Test-CFNTemplate

Strumenti per PowerShell

Esempio 1: convalida il contenuto del modello specificato. L'output descrive in dettaglio le funzionalità, la descrizione e i parametri del modello.

```
Test-CFNTemplate -TemplateBody "{TEMPLATE CONTENT HERE}"
```

Esempio 2: convalida il modello specificato a cui si accede tramite un URL Amazon S3. L'output descrive in dettaglio le funzionalità, la descrizione e i parametri del modello.

```
Test-CFNTemplate -TemplateURL https://s3.amazonaws.com/mytemplates/
templatefile.template
```

- Per i dettagli sull'API, vedere [ValidateTemplate](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-CFNStack

Il seguente esempio di codice mostra come utilizzare. Update-CFNStack

Strumenti per PowerShell

Esempio 1: aggiorna lo stack 'MyStack' con il modello e i parametri di personalizzazione specificati. 'PK1' rappresenta il nome di un parametro dichiarato nel modello e 'PV1' ne rappresenta il valore. I parametri di personalizzazione possono essere specificati anche utilizzando «Key» e «Value» anziché «>>» e «<<». ParameterKey ParameterValue

```
Update-CFNStack -StackName "myStack" `
  -TemplateBody "{Template Content Here}" `
  -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Esempio 2: aggiorna lo stack 'MyStack' con il modello e i parametri di personalizzazione specificati. 'PK1' e 'PK2' rappresentano i nomi dei parametri dichiarati nel modello, 'PV1' e 'PV2' rappresentano i valori richiesti. I parametri di personalizzazione possono anche essere specificati utilizzando «Key» e «Value» anziché «>>» e «<<». ParameterKey ParameterValue

```
Update-CFNStack -StackName "myStack" `
  -TemplateBody "{Template Content Here}" `
  -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
  @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

Esempio 3: aggiorna lo stack 'MyStack' con il modello e i parametri di personalizzazione specificati. 'PK1' rappresenta il nome di un parametro dichiarato nel modello e 'PV2' ne rappresenta il valore. I parametri di personalizzazione possono anche essere specificati utilizzando «Key» e «Value» anziché «>>» e «<<». ParameterKey ParameterValue


```
Update-CFNStack -StackName "myStack" -TemplateBody "{Template Content Here}" -
Parameters @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Esempio 4: aggiorna lo stack 'MyStack' con il modello specificato, ottenuto da Amazon S3, e i parametri di personalizzazione. «PK1» e «PK2» rappresentano i nomi dei parametri dichiarati nel modello, «PV1» e «PV2» rappresentano i valori richiesti. I parametri di personalizzazione possono anche essere specificati utilizzando «Key» e «Value» anziché «» e «». ParameterKey ParameterValue

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/mytemplates/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
@{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

Esempio 5: aggiorna lo stack 'MyStack', che in questo esempio si presume contenga risorse IAM, con il modello specificato, ottenuto da Amazon S3, e i parametri di personalizzazione. «PK1» e «PK2» rappresentano i nomi dei parametri dichiarati nel modello, «PV1» e «PV2» rappresentano i valori richiesti. I parametri di personalizzazione possono anche essere specificati utilizzando «Key» e «Value» anziché «» e «». ParameterKey ParameterValue Gli stack contenenti risorse IAM richiedono di specificare il parametro -Capabilities «CAPABILITY_IAM», altrimenti l'aggiornamento avrà esito negativo con un errore ". 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"
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [UpdateStack](#)AWS Tools for PowerShell

CloudFront esempi di utilizzo di Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with CloudFront.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Get-CFCloudFrontOriginAccessIdentity

Il seguente esempio di codice mostra come utilizzare `Get-CFCloudFrontOriginAccessIdentity`.

Strumenti per PowerShell

Esempio 1: questo esempio restituisce un'identità di accesso all' CloudFront origine di Amazon specifica, specificata dal parametro `-Id`. Sebbene il parametro `-Id` non sia obbligatorio, se non lo si specifica non viene restituito alcun risultato.

```
Get-CFCloudFrontOriginAccessIdentity -Id E3XXXXXXXXXXRT
```

Output:

```
CloudFrontOriginAccessIdentityConfig    Id
S3CanonicalUserId
-----
Amazon.CloudFront.Model.CloudFrontOr... E3XXXXXXXXXXRT
4b6e...
```

- Per i dettagli sull'API, vedere [GetCloudFrontOriginAccessIdentity](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CFCloudFrontOriginAccessIdentityConfig

Il seguente esempio di codice mostra come utilizzare. Get - CFCloudFrontOriginAccessIdentityConfig

Strumenti per PowerShell

Esempio 1: questo esempio restituisce informazioni di configurazione su una singola identità di accesso di CloudFront origine Amazon, specificata dal parametro -Id. Si verificano errori se non viene specificato alcun parametro -Id.

```
Get-CFCloudFrontOriginAccessIdentityConfig -Id E3XXXXXXXXXXRT
```

Output:

CallerReference	Comment
-----	-----
mycallerreference: 2/1/2011 1:16:32 PM 2/1/2011 1:16:32 PM	Caller reference:

- Per i dettagli sull'API, vedere [GetCloudFrontOriginAccessIdentityConfig](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CFCloudFrontOriginAccessIdentityList

Il seguente esempio di codice mostra come utilizzare. Get - CFCloudFrontOriginAccessIdentityList

Strumenti per PowerShell

Esempio 1: questo esempio restituisce un elenco di identità di accesso di CloudFront origine di Amazon. Poiché il MaxItem parametro - specifica il valore 2, i risultati includono due identità.

```
Get-CFCloudFrontOriginAccessIdentityList -MaxItem 2
```

Output:

```
IsTruncated : True
Items       : {E326XXXXXXXXXXT, E1YWXXXXXXXX9B}
```

```
Marker      :  
MaxItems    : 2  
NextMarker  : E1YXXXXXXXXXX9B  
Quantity    : 2
```

- Per i dettagli sull'API, vedere [ListCloudFrontOriginAccessIdentities](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CFDistribution

Il seguente esempio di codice mostra come utilizzare. `Get-CFDistribution`

Strumenti per PowerShell

Esempio 1: recupera le informazioni per una distribuzione specifica.

```
Get-CFDistribution -Id EXAMPLE0000ID
```

- Per i dettagli sull'API, vedere [GetDistribution](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CFDistributionConfig

Il seguente esempio di codice mostra come utilizzare. `Get-CFDistributionConfig`

Strumenti per PowerShell

Esempio 1: recupera la configurazione per una distribuzione specifica.

```
Get-CFDistributionConfig -Id EXAMPLE0000ID
```

- Per i dettagli sull'API, vedere [GetDistributionConfig](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CFDistributionList

Il seguente esempio di codice mostra come utilizzare. `Get-CFDistributionList`

Strumenti per PowerShell

Esempio 1: restituisce le distribuzioni.

Get-CFDistributionList

- Per i dettagli sull'API, vedere [ListDistributions](#) in AWS Tools for PowerShell Cmdlet Reference.

New-CFDistribution

Il seguente esempio di codice mostra come utilizzare `New-CFDistribution`

Strumenti per PowerShell

Esempio 1: crea una CloudFront distribuzione di base, configurata con registrazione e memorizzazione nella cache.

```
$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
```

- Per i dettagli sull'API, vedere [CreateDistribution](#) in AWS Tools for PowerShell Cmdlet Reference.

New-CFInvalidation

Il seguente esempio di codice mostra come utilizzare `New-CFInvalidation`

Strumenti per PowerShell

Esempio 1: questo esempio crea una nuova invalidazione su una distribuzione con un ID `EXAMPLENSTXAXE`. `CallerReference` è un ID univoco scelto dall'utente; in questo caso, viene utilizzato un timestamp che rappresenta il 15 maggio 2019 alle 9:00. La variabile `$Paths` memorizza tre percorsi di immagini e file multimediali che l'utente non desidera vengano inseriti nella cache della distribuzione. Il valore del parametro `-Paths_Quantity` è il numero totale di percorsi specificati nel parametro `-Paths_Item`.

```
$Paths = "/images/*.gif", "/images/image1.jpg", "/videos/*.mp4"
New-CFInvalidation -DistributionId "EXAMPLENSTXAXE" -
InvalidationBatch_CallerReference 20190515090000 -Paths_Item $Paths -Paths_Quantity
3
```

Output:

```
Invalidation          Location
-----
Amazon.CloudFront.Model.Invalidation https://cloudfront.amazonaws.com/2018-11-05/
distribution/EXAMPLENSTXAXE/invalidation/EXAMPLE8N0K9H
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [CreateInvalidation](#) AWS Tools for PowerShell

New-CFSignedCookie

Il seguente esempio di codice mostra come utilizzare `New-CFSignedCookie`

Strumenti per PowerShell

Esempio 1: crea un cookie firmato per la risorsa specificata utilizzando una politica predefinita. Il cookie sarà valido per un anno.

```
$params = @{
```

```
"ResourceUri"="http://xyz.cloudfront.net/image1.jpeg"
"KeyPairId"="AKIAIOSFODNN7EXAMPLE"
"PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
"ExpiresOn"=(Get-Date).AddYears(1)
}
New-CFSignedCookie @params
```

Output:

```
Expires
-----
[CloudFront-Expires, 1472227284]
```

Esempio 2: crea un cookie firmato per le risorse specificate utilizzando una politica personalizzata. Il cookie sarà valido tra 24 ore e scadrà una settimana dopo.

```
$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
```

Output:

```
Policy
-----
[CloudFront-Policy, eyJTd...wIjo...
```

Esempio 3: crea un cookie firmato per le risorse specificate utilizzando una politica personalizzata. Il cookie sarà valido tra 24 ore e scadrà una settimana dopo. L'accesso alle risorse è limitato all'intervallo IP specificato.

```
$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
```

Output:

```
Policy
-----
[CloudFront-Policy, eyJTd...wIjo...
```

- Per i dettagli sull'API, vedere [New-CF SignedCookie in AWS Tools for PowerShell Cmdlet Reference](#).

New-CFSignedUrl

Il seguente esempio di codice mostra come utilizzare `New-CFSignedUrl`

Strumenti per PowerShell

Esempio 1: crea un URL firmato per la risorsa specificata utilizzando una politica predefinita. L'URL sarà valido per un'ora. Un oggetto `System.Uri` contenente l'URL firmato viene emesso nella pipeline.

```
$params = @{
  "ResourceUri"="https://cdn.example.com/index.html"
  "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
  "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
  "ExpiresOn"=(Get-Date).AddHours(1)
}

New-CFSignedUrl @params
```

Esempio 2: crea un URL firmato per la risorsa specificata utilizzando una politica personalizzata. L'URL sarà valido a partire da 24 ore e scadrà una settimana dopo.

```
$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
```

Esempio 3: crea un URL firmato per la risorsa specificata utilizzando una politica personalizzata. L'URL sarà valido a partire da 24 ore e scadrà una settimana dopo. L'accesso alla risorsa è limitato all'intervallo IP specificato.

```
$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
```

- Per i dettagli sull'API, vedere [New-CF SignedUrl in AWS Tools for PowerShell Cmdlet Reference](#).

CloudTrail esempi di utilizzo di Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with CloudTrail.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Find-CTEvent

Il seguente esempio di codice mostra come utilizzare Find-CTEvent.

Strumenti per PowerShell

Esempio 1: restituisce tutti gli eventi che si sono verificati negli ultimi sette giorni. Per impostazione predefinita, il cmdlet effettua automaticamente più chiamate per fornire tutti gli eventi, uscendo quando il servizio indica che non sono disponibili altri dati.

```
Find-CTEvent
```

Esempio 2: restituisce tutti gli eventi che si sono verificati negli ultimi sette giorni specificando un'area che non è l'impostazione predefinita della shell corrente.

```
Find-CTEvent -Region eu-central-1
```

Esempio 3: restituisce tutti gli eventi associati alla chiamata RunInstances API.

```
Find-CTEvent -LookupAttribute @{ AttributeKey="EventName";  
  AttributeValue="RunInstances" }
```

Esempio 4: restituisce i primi 5 eventi disponibili. Il token da utilizzare per recuperare ulteriori eventi è allegato come proprietà nota denominata 'NextToken' al **\$AWSHistory.LastServiceResponse** membro.

```
Find-CTEvent -MaxResult 5
```

Esempio 5: restituisce i 10 eventi successivi utilizzando il token 'next page' di una chiamata precedente per indicare da dove iniziare a restituire gli eventi della sequenza.

```
Find-CTEvent -MaxResult 10 -NextToken $AWSHistory.LastServiceResponse.NextToken
```

Esempio 6: Questo esempio mostra come scorrere gli eventi disponibili utilizzando il paging manuale, recuperando un massimo di 5 eventi per chiamata.

```
$nextToken = $null
do
{
    Find-CTEvent -MaxResult 5 -NextToken $nextToken
    $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Per i dettagli sull'API, vedere [LookupEvents](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CTTrail

Il seguente esempio di codice mostra come utilizzare `Get-CTTrail`

Strumenti per PowerShell

Esempio 1: restituisce le impostazioni di tutti i percorsi associati alla regione corrente del tuo account.

```
Get-CTTrail
```

Esempio 2: restituisce le impostazioni per i percorsi specificati.

```
Get-CTTrail -TrailNameList trail1, trail2
```

Esempio 3: restituisce le impostazioni per i percorsi specificati che sono stati creati in una regione diversa dall'attuale impostazione predefinita della shell (in questo caso la regione di Francoforte (eu-central-1)).

```
Get-CTTrail -TrailNameList trailABC, trailDEF -Region eu-central-1
```

- Per i dettagli sull'API, vedere [DescribeTrails](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CTTrailStatus

Il seguente esempio di codice mostra come utilizzare `Get-CTTrailStatus`

Strumenti per PowerShell

Esempio 1: restituisce informazioni sullo stato del percorso con il nome 'myExampleTrail'. I dati restituiti includono informazioni sugli errori di consegna, sugli errori di Amazon SNS e Amazon S3 e sui tempi di avvio e interruzione della registrazione per il percorso. Questo esempio presuppone che il trail sia stato creato nella stessa regione della shell predefinita corrente.

```
Get-CTTrailStatus -Name myExampleTrail
```

Esempio 2: restituisce le informazioni sullo stato di un percorso creato in una regione diversa da quella predefinita della shell corrente (in questo caso, la regione di Francoforte (eu-central-1)).

```
Get-CTTrailStatus -Name myExampleTrail -Region eu-central-1
```

- Per i dettagli sull'API, vedere [GetTrailStatus](#) in AWS Tools for PowerShell Cmdlet Reference.

New-CTTrail

Il seguente esempio di codice mostra come utilizzare `New-CTTrail`

Strumenti per PowerShell

Esempio 1: crea un percorso che utilizzerà il bucket 'mycloudtrailbucket' per l'archiviazione dei file di registro.

```
New-CTTrail -Name="awscloudtrail-example" -S3BucketName="mycloudtrailbucket"
```

Esempio 2: crea un percorso che utilizzerà il bucket 'mycloudtrailbucket' per l'archiviazione dei file di registro. Gli oggetti S3 che rappresentano i log avranno un prefisso chiave comune di 'mylogs'. Quando vengono consegnati nuovi log al bucket, verrà inviata una notifica all'argomento SNS 'mlog-deliverytopic'. In questo esempio viene utilizzato lo splatting per fornire i valori dei parametri al cmdlet.

```
$params = @{  
    Name="awscloudtrail-example"  
    S3BucketName="mycloudtrailbucket"  
    S3KeyPrefix="mylogs"  
    SnsTopicName="mlog-deliverytopic"  
}
```

```
New-CTTrail @params
```

- Per i dettagli sull'API, vedere [CreateTrail](#) in Cmdlet Reference.AWS Tools for PowerShell

Remove-CTTrail

Il seguente esempio di codice mostra come utilizzare. `Remove-CTTrail`

Strumenti per PowerShell

Esempio 1: elimina la traccia specificata. Ti verrà richiesta una conferma prima dell'esecuzione del comando. Per eliminare la conferma, aggiungi il parametro `-Force` switch.

```
Remove-CTTrail -Name "awscloudtrail-example"
```

- Per i dettagli sull'API, vedere [DeleteTrail](#) in AWS Tools for PowerShell Cmdlet Reference.

Start-CTLogging

Il seguente esempio di codice mostra come utilizzare. `Start-CTLogging`

Strumenti per PowerShell

Esempio 1: avvia la registrazione delle chiamate AWS API e la consegna dei file di registro per il percorso denominato 'myExampleTrail'. Questo esempio presuppone che il trail sia stato creato nella stessa regione della shell predefinita corrente.

```
Start-CTLogging -Name myExampleTrail
```

Esempio 2: avvia la registrazione delle chiamate AWS API e la consegna dei file di registro per un percorso creato in una regione diversa dall'attuale impostazione predefinita della shell (in questo caso, la regione di Francoforte (eu-central-1)).

```
Start-CTLogging -Name myExampleTrail -Region eu-central-1
```

- Per i dettagli sull'API, vedere [StartLogging](#) in AWS Tools for PowerShell Cmdlet Reference.

Stop-CTLogging

Il seguente esempio di codice mostra come utilizzare. `Stop-CTLogging`

Strumenti per PowerShell

Esempio 1: sospende la registrazione delle chiamate AWS API e la consegna dei file di registro per il percorso denominato 'myExampleTrail'. Questo esempio presuppone che il trail sia stato creato nella stessa regione della shell predefinita corrente.

```
Stop-CTLogging -Name myExampleTrail
```

Esempio 2: sospende la registrazione delle chiamate AWS API e la consegna dei file di registro per un trail creato in una regione diversa dall'attuale impostazione predefinita della shell (in questo caso, la regione di Francoforte (eu-central-1)).

```
Stop-CTLogging -Name myExampleTrail -Region eu-central-1
```

- Per i dettagli sull'API, vedere [StopLogging](#) in Cmdlet Reference.AWS Tools for PowerShell

Update-CTTrail

Il seguente esempio di codice mostra come utilizzare. Update-CTTrail

Strumenti per PowerShell

Esempio 1: aggiorna il percorso specificato in modo che gli eventi del servizio globale (come quelli di IAM) vengano registrati e modifica il prefisso della chiave comune dei file di registro in futuro in «globallogs».

```
Update-CTTrail -Name "awscloudtrail-example" -IncludeGlobalServiceEvents $true -  
S3KeyPrefix "globallogs"
```

Esempio 2: aggiorna il percorso specificato in modo che le notifiche sulle nuove consegne di log vengano inviate all'argomento SNS specificato.

```
Update-CTTrail -Name "awscloudtrail-example" -SnsTopicName "mlog-deliverytopic2"
```

Esempio 3: aggiorna il percorso specificato in modo che i log vengano recapitati a un bucket diverso.

```
Update-CTTrail -Name "awscloudtrail-example" -S3BucketName "otherlogs"
```

- Per i dettagli sull'API, vedere [UpdateTrail](#) in AWS Tools for PowerShell Cmdlet Reference.

CloudWatch esempi di utilizzo di Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with CloudWatch.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Get-CWDashboard

Il seguente esempio di codice mostra come utilizzare `Get-CWDashboard`.

Strumenti per PowerShell

Esempio 1: restituisce l'arn al corpo della dashboard specificata.

```
Get-CWDashboard -DashboardName Dashboard1
```

Output:

```
DashboardArn          DashboardBody
-----
arn:aws:cloudwatch::123456789012:dashboard/Dashboard1 {...
```

- Per i dettagli sull'API, vedere [GetDashboard](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CWDashboardList

Il seguente esempio di codice mostra come utilizzare. `Get-CWDashboardList`

Strumenti per PowerShell

Esempio 1: restituisce la raccolta di dashboard per il tuo account.

```
Get-CWDashboardList
```

Output:

```
DashboardArn DashboardName LastModified      Size
-----
arn:...      Dashboard1    7/6/2017 8:14:15 PM 252
```

Esempio 2: restituisce la raccolta di dashboard per il tuo account i cui nomi iniziano con il prefisso 'dev'.

```
Get-CWDashboardList -DashboardNamePrefix dev
```

- Per i dettagli sull'API, vedere [ListDashboards](#) in Cmdlet Reference.AWS Tools for PowerShell

Remove-CWDashboard

Il seguente esempio di codice mostra come utilizzare. `Remove-CWDashboard`

Strumenti per PowerShell

Esempio 1: elimina la dashboard specificata, richiedendone conferma prima di procedere. Per ignorare la conferma, aggiungi l'interruttore `-Force` al comando.

```
Remove-CWDashboard -DashboardName Dashboard1
```

- Per i dettagli sull'API, vedere [DeleteDashboards](#) in AWS Tools for PowerShell Cmdlet Reference.

Write-CWDashboard

Il seguente esempio di codice mostra come utilizzare. `Write-CWDashboard`

Strumenti per PowerShell

Esempio 1: crea o aggiorna la dashboard denominata «Dashboard1» per includere due widget metriche affiancati.

```
$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

```

Esempio 2: crea o aggiorna il dashboard, inserendo il contenuto che descrive il dashboard nel cmdlet.

```

$dashBody = @"
{
...
}
"@

$dashBody | Write-CWDashboard -DashboardName Dashboard1

```

- Per i dettagli sull'API, vedere [PutDashboard](#) in Cmdlet Reference.AWS Tools for PowerShell

Write-CWMetricData

Il seguente esempio di codice mostra come utilizzare `Write-CWMetricData`

Strumenti per PowerShell

Esempio 1: crea un nuovo `MetricDatum` oggetto e lo scrive su 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

```

- Per i dettagli sull'API, consulta AWS Tools for PowerShell Cmdlet [PutMetricDataReference](#).

CodeCommit esempi di utilizzo di Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with CodeCommit.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Get-CCBranch

Il seguente esempio di codice mostra come utilizzare `Get-CCBranch`.

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene informazioni sul ramo specificato per il repository specificato.

```
Get-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch
```

Output:

BranchName	CommitId
-----	-----
MyNewBranch	7763222d...561fc9c9

- Per i dettagli sull'API, vedere [GetBranchin](#) AWS Tools for PowerShell Cmdlet Reference.

Get-CCBranchList

Il seguente esempio di codice mostra come utilizzare. `Get-CCBranchList`

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene un elenco di nomi di ramo per il repository specificato.

```
Get-CCBranchList -RepositoryName MyDemoRepo
```

Output:

```
master
MyNewBranch
```

- Per i dettagli sull'API, vedere [ListBranches](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CCRepository

Il seguente esempio di codice mostra come utilizzare. `Get-CCRepository`

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene informazioni per il repository specificato.

```
Get-CCRepository -RepositoryName MyDemoRepo
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [GetRepository](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CCRepositoryBatch

Il seguente esempio di codice mostra come utilizzare. Get-CCRepositoryBatch

Strumenti per PowerShell

Esempio 1: Questo esempio conferma quali dei repository specificati sono stati trovati e non trovati.

```
Get-CCRepositoryBatch -RepositoryName MyDemoRepo, MyNewRepo, AMissingRepo
```

Output:

Repositories	RepositoriesNotFound
-----	-----
{MyDemoRepo, MyNewRepo}	{AMissingRepo}

- Per i dettagli sull'API, vedere [BatchGetRepositories](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CCRepositoryList

Il seguente esempio di codice mostra come utilizzare. Get-CCRepositoryList

Strumenti per PowerShell

Esempio 1: Questo esempio elenca tutti i repository in ordine crescente in base al nome del repository.

```
Get-CCRepositoryList -Order Ascending -SortBy RepositoryName
```

Output:

RepositoryId	RepositoryName
-----	-----
c7d0d2b0-ce40-4303-b4c3-38529EXAMPLE	MyDemoRepo
05f30c66-e3e3-4f91-a0cd-1c84aEXAMPLE	MyNewRepo

- Per i dettagli sull'API, vedere [ListRepositories](#) in AWS Tools for PowerShell Cmdlet Reference.

New-CCBranch

Il seguente esempio di codice mostra come utilizzare. New-CCBranch

Strumenti per PowerShell

Esempio 1: Questo esempio crea un nuovo ramo con il nome specificato per il repository specificato e l'ID di commit specificato.

```
New-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch -CommitId
7763222d...561fc9c9
```

- Per i dettagli sull'API, vedere [CreateBranch](#) in AWS Tools for PowerShell Cmdlet Reference.

New-CCRepository

Il seguente esempio di codice mostra come utilizzare. New-CCRepository

Strumenti per PowerShell

Esempio 1: Questo esempio crea un nuovo repository con il nome e la descrizione specificati.

```
New-CCRepository -RepositoryName MyDemoRepo -RepositoryDescription "This is a
repository for demonstration purposes."
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [CreateRepository](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-CCRepository

Il seguente esempio di codice mostra come utilizzare. Remove-CCRepository

Strumenti per PowerShell

Esempio 1: questo esempio elimina forzatamente il repository specificato. Il comando richiederà una conferma prima di procedere. Aggiungi il parametro -Force per eliminare il repository senza una richiesta.

```
Remove-CCRepository -RepositoryName MyDemoRepo
```

Output:

```
43ef2443-3372-4b12-9e78-65c27EXAMPLE
```

- Per i dettagli sull'API, vedere [DeleteRepository](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-CCDefaultBranch

Il seguente esempio di codice mostra come utilizzare. Update-CCDefaultBranch

Strumenti per PowerShell

Esempio 1: Questo esempio modifica il ramo predefinito per il repository specificato nel ramo specificato.

```
Update-CCDefaultBranch -RepositoryName MyDemoRepo -DefaultBranchName MyNewBranch
```

- Per i dettagli sull'API, vedere [UpdateDefaultBranch](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-CCRepositoryDescription

Il seguente esempio di codice mostra come utilizzare. Update-CCRepositoryDescription

Strumenti per PowerShell

Esempio 1: questo esempio modifica la descrizione del repository specificato.

```
Update-CCRepositoryDescription -RepositoryName MyDemoRepo -RepositoryDescription  
"This is an updated description."
```

- Per i dettagli sull'API, vedere [UpdateRepositoryDescription](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-CCRepositoryName

Il seguente esempio di codice mostra come utilizzare. Update-CCRepositoryName

Strumenti per PowerShell

Esempio 1: Questo esempio modifica il nome del repository specificato.

```
Update-CCRepositoryName -NewName MyDemoRepo2 -OldName MyDemoRepo
```

- Per i dettagli sull'API, vedere [UpdateRepositoryName](#) in AWS Tools for PowerShell Cmdlet Reference.

CodeDeploy esempi di utilizzo di Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with CodeDeploy.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Add-CDOnPremiseInstanceTag

Il seguente esempio di codice mostra come utilizzare `Add-CDOnPremiseInstanceTag`.

Strumenti per PowerShell

Esempio 1: Questo esempio aggiunge un tag di istanza locale con la chiave e il valore specificati per l'istanza locale specificata.

```
Add-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" = "Name";  
"Value" = "CodeDeployDemo-OnPrem"}
```

- Per i dettagli sull'API, vedere [AddTagsToOnPremisesInstances](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CDApplication

Il seguente esempio di codice mostra come utilizzare `Get-CDApplication`.

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene informazioni sull'applicazione specificata.

```
Get-CDApplication -ApplicationName CodeDeployDemoApplication
```

Output:

ApplicationId	ApplicationName	CreateTime
----- ----- LinkedToGitHub	-----	-----
e07fb938-091e-4f2f-8963-4d3e8EXAMPLE 9:49:48 PM False	CodeDeployDemoApplication	7/20/2015

- Per i dettagli sull'API, vedere [GetApplication](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CDApplicationBatch

Il seguente esempio di codice mostra come utilizzare `Get-CDApplicationBatch`.

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene informazioni sulle applicazioni specificate.

```
Get-CDApplicationBatch -ApplicationName CodeDeployDemoApplication,
CodePipelineDemoApplication
```

Output:

ApplicationId	ApplicationName	CreateTime
LinkedToGitHub ----- -----	-----	-----
e07fb938-091e-4f2f-8963-4d3e8EXAMPLE 9:49:48 PM False	CodeDeployDemoApplication	7/20/2015
1ecfd602-62f1-4038-8f0d-06688EXAMPLE 5:53:26 PM False	CodePipelineDemoApplication	8/13/2015

- Per i dettagli sull'API, vedere [BatchGetApplications](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CDApplicationList

Il seguente esempio di codice mostra come utilizzare. Get-CDApplicationList

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene un elenco di applicazioni disponibili.

```
Get-CDApplicationList
```

Output:

```
CodeDeployDemoApplication
CodePipelineDemoApplication
```

- Per i dettagli sull'API, vedere [ListApplications](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CDApplicationRevision

Il seguente esempio di codice mostra come utilizzare. Get-CDApplicationRevision

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene informazioni sulla revisione dell'applicazione specificata.

```
$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)
```

Output:

```
Description = Application revision registered by Deployment ID: d-CX9CHN3EX,
RegisterTime = 07/20/2015 23:46:42
```

- Per i dettagli sull'API, vedere [GetApplicationRevision](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CDApplicationRevisionList

Il seguente esempio di codice mostra come utilizzare. Get-CDApplicationRevisionList

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene informazioni sulle revisioni disponibili per l'applicazione specificata.

```
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)
>> }
>> }
>> }
```

Output:

```
Type = S3, Bucket = MyBucket, BundleType = zip, ETag =
4565c1ac97187f190c1a90265EXAMPLE, Key = 5xd27EX.zip
Type = GitHub, CommitId = f48933c3...76405362, Repository = MyGitHubUser/
CodeDeployDemoRepo
```

- Per i dettagli sull'API, vedere [ListApplicationRevisions](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CDDeployment

Il seguente esempio di codice mostra come utilizzare Get-CDDeployment

Strumenti per PowerShell

Esempio 1: questo esempio ottiene informazioni di riepilogo sulla distribuzione specificata.

```
Get-CDDeployment -DeploymentId d-QZMRGSTEX
```

Output:

```
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
```

Esempio 2: questo esempio ottiene informazioni sullo stato delle istanze che partecipano alla distribuzione specificata.

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).DeploymentOverview
```

Output:

```
Failed      : 0
InProgress  : 0
Pending     : 0
Skipped     : 0
Succeeded   : 3
```

Esempio 3: questo esempio ottiene informazioni sulla revisione dell'applicazione per la distribuzione specificata.

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).Revision.S3Location
```

Output:

```
Bucket      : MyBucket
BundleType  : zip
ETag        : cfbb81b304ee5e27efc21adaed3EXAMPLE
Key         : clzfqEX
Version     :
```

- Per i dettagli sull'API, vedere [GetDeployment](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CDDeploymentBatch

Il seguente esempio di codice mostra come utilizzare `Get-CDDeploymentBatch`

Strumenti per PowerShell

Esempio 1: questo esempio ottiene informazioni sulle distribuzioni specificate.

```
Get-CDDeploymentBatch -DeploymentId d-QZMRGSTEX, d-RR0T5KTEX
```

Output:

```
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

```

- Per i dettagli sull'API, vedere [BatchGetDeployments](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CDDeploymentConfig

Il seguente esempio di codice mostra come utilizzare. Get-CDDeploymentConfig

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene informazioni di riepilogo sulla configurazione di distribuzione specificata.

```
Get-CDDeploymentConfig -DeploymentConfigName ThreeQuartersHealthy
```

Output:

```

CreateTime           DeploymentConfigId           DeploymentConfigName
-----
MinimumHealthyHosts

```

```
-----  
-----  
-----  
10/3/2014 4:32:30 PM 518a3950-d034-46a1-9d2c-3c949EXAMPLE ThreeQuartersHealthy  
Amazon.CodeDeploy.Model.MinimumHealthyHosts
```

Esempio 2: Questo esempio ottiene informazioni sulla definizione della configurazione di distribuzione specificata.

```
Write-Output ((Get-CDDeploymentConfig -DeploymentConfigName  
ThreeQuartersHealthy).MinimumHealthyHosts)
```

Output:

Type	Value
----	----
FLEET_PERCENT	75

- Per i dettagli sull'API, vedere [GetDeploymentConfig](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CDDeploymentConfigList

Il seguente esempio di codice mostra come utilizzare `Get-CDDeploymentConfigList`

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene un elenco di configurazioni di distribuzione disponibili.

```
Get-CDDeploymentConfigList
```

Output:

```
ThreeQuartersHealthy  
CodeDeployDefault.OneAtATime  
CodeDeployDefault.AllAtOnce  
CodeDeployDefault.HalfAtATime
```

- Per i dettagli sull'API, vedere [ListDeploymentConfigs](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CDDeploymentGroup

Il seguente esempio di codice mostra come utilizzare. `Get-CDDeploymentGroup`

Strumenti per PowerShell

Esempio 1: questo esempio ottiene informazioni sul gruppo di distribuzione specificato.

```
Get-CDDeploymentGroup -ApplicationName CodeDeployDemoApplication -  
DeploymentGroupName CodeDeployDemoFleet
```

Output:

```
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-09MWP7XTL8EX  
TargetRevision           : Amazon.CodeDeploy.Model.RevisionLocation
```

- Per i dettagli sull'API, vedere [GetDeploymentGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CDDeploymentGroupList

Il seguente esempio di codice mostra come utilizzare. `Get-CDDeploymentGroupList`

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene un elenco di gruppi di distribuzione per l'applicazione specificata.

```
Get-CDDeploymentGroupList -ApplicationName CodeDeployDemoApplication
```

Output:

```
ApplicationName           DeploymentGroups  
NextToken
```



```

-----
-----
CodeDeployDemoApplication    {CodeDeployDemoFleet, CodeDeployProductionFleet}

```

- Per i dettagli sull'API, vedere [ListDeploymentGroups](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CDDeploymentInstance

Il seguente esempio di codice mostra come utilizzare. `Get-CDDeploymentInstance`

Strumenti per PowerShell

Esempio 1: questo esempio ottiene informazioni sull'istanza specificata per la distribuzione specificata.

```
Get-CDDeploymentInstance -DeploymentId d-QZMRGSTEX -InstanceId i-254e22EX
```

Output:

```

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

```

- Per i dettagli sull'API, vedere [GetDeploymentInstance](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CDDeploymentInstanceList

Il seguente esempio di codice mostra come utilizzare. `Get-CDDeploymentInstanceList`

Strumenti per PowerShell

Esempio 1: questo esempio ottiene un elenco di ID di istanza per la distribuzione specificata.

```
Get-CDDeploymentInstanceList -DeploymentId d-QZMRGSTEX
```

Output:

```
i-254e22EX  
i-274e22EX  
i-3b4e22EX
```

- Per i dettagli sull'API, vedere [ListDeploymentInstances](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CDDeploymentList

Il seguente esempio di codice mostra come utilizzare. `Get-CDDeploymentList`

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene un elenco di ID di distribuzione per l'applicazione e il gruppo di distribuzione specificati.

```
Get-CDDeploymentList -ApplicationName CodeDeployDemoApplication -DeploymentGroupName  
CodeDeployDemoFleet
```

Output:

```
d-QZMRGSTEX  
d-RR0T5KTEX
```

- Per i dettagli sull'API, vedere [ListDeployments](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CDOnPremiseInstance

Il seguente esempio di codice mostra come utilizzare. `Get-CDOnPremiseInstance`

Strumenti per PowerShell

Esempio 1: questo esempio ottiene informazioni sull'istanza locale specificata.

```
Get-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

Output:

```
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}
```

- Per i dettagli sull'API, vedere [GetOnPremisesInstance](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CDOnPremiseInstanceBatch

Il seguente esempio di codice mostra come utilizzare. Get-CDOnPremiseInstanceBatch

Strumenti per PowerShell

Esempio 1: questo esempio ottiene informazioni sulle istanze locali specificate.

```
Get-CDOnPremiseInstanceBatch -InstanceName AssetTag12010298EX, AssetTag12010298EX-2
```

Output:

```
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}
```

- Per i dettagli sull'API, vedere [BatchGetOnPremisesInstances](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CDOnPremiseInstanceList

Il seguente esempio di codice mostra come utilizzare. Get-CDOnPremiseInstanceList

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene un elenco di nomi di istanze locali disponibili.

```
Get-CDOnPremiseInstanceList
```

Output:

```
AssetTag12010298EX  
AssetTag12010298EX-2
```

- Per i dettagli sull'API, vedere [ListOnPremisesInstances](#) in AWS Tools for PowerShell Cmdlet Reference.

New-CDApplication

Il seguente esempio di codice mostra come utilizzare. New-CDApplication

Strumenti per PowerShell

Esempio 1: Questo esempio crea una nuova applicazione con il nome specificato.

```
New-CDApplication -ApplicationName MyNewApplication
```

Output:

```
f19e4b61-2231-4328-b0fd-e57f5EXAMPLE
```

- Per i dettagli sull'API, vedere [CreateApplication](#) in AWS Tools for PowerShell Cmdlet Reference.

New-CDDeployment

Il seguente esempio di codice mostra come utilizzare. New-CDDeployment

Strumenti per PowerShell

Esempio 1: Questo esempio crea una nuova distribuzione per l'applicazione e il gruppo di distribuzione specificati con la configurazione di distribuzione e la revisione dell'applicazione specificate.

```
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
```

Output:

```
d-ZHR0G7UEX
```

Esempio 2: Questo esempio mostra come specificare i gruppi di tag di istanza EC2 con cui un'istanza deve essere identificata per poter essere inclusa nell'ambiente sostitutivo per una distribuzione blu/verde.

```
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=
```

Output:

```
d-ZHR0G7UEX
```

- Per i dettagli sull'API, vedere [CreateDeployment](#) in AWS Tools for PowerShell Cmdlet Reference.

New-CDDeploymentConfig

Il seguente esempio di codice mostra come utilizzare `New-CDDeploymentConfig`

Strumenti per PowerShell

Esempio 1: questo esempio crea una nuova configurazione di distribuzione con il nome e il comportamento specificati.

```
New-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts -
MinimumHealthyHosts_Type HOST_COUNT -MinimumHealthyHosts_Value 2
```

Output:

```
0f3e8187-44ef-42da-aeed-b6823EXAMPLE
```

- Per i dettagli sull'API, vedere [CreateDeploymentConfig](#) in AWS Tools for PowerShell Cmdlet Reference.

New-CDDeploymentGroup

Il seguente esempio di codice mostra come utilizzare `New-CDDeploymentGroup`

Strumenti per PowerShell

Esempio 1: questo esempio crea un gruppo di distribuzione con il nome specificato, gruppo Auto Scaling, configurazione di distribuzione, tag e ruolo di servizio, per l'applicazione specificata.

```
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
```

Output:

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

Esempio 2: questo esempio mostra come specificare i gruppi di tag di istanza EC2 in base ai quali un'istanza deve essere identificata per poter essere inclusa nell'ambiente sostitutivo di una distribuzione blu/verde.

```
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=
```

Output:

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

- Per i dettagli sull'API, vedere [CreateDeploymentGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

Register-CDApplicationRevision

Il seguente esempio di codice mostra come utilizzare. Register-CDApplicationRevision

Strumenti per PowerShell

Esempio 1: questo esempio registra una revisione dell'applicazione con la posizione Amazon S3 specificata, per l'applicazione specificata.

```
Register-CDApplicationRevision -ApplicationName MyNewApplication -S3Location_Bucket  
MyBucket -S3Location_BundleType zip -S3Location_Key aws-codedeploy_linux-master.zip  
-Revision_RevisionType S3
```

- Per i dettagli sull'API, vedere [RegisterApplicationRevision](#) in AWS Tools for PowerShell Cmdlet Reference.

Register-CDOnPremiseInstance

Il seguente esempio di codice mostra come utilizzare. Register-CDOnPremiseInstance

Strumenti per PowerShell

Esempio 1: questo esempio registra un'istanza locale con il nome e l'utente IAM specificati.

```
Register-CDOnPremiseInstance -IamUserArn arn:aws:iam::80398EXAMPLE:user/  
CodeDeployDemoUser -InstanceName AssetTag12010298EX
```

- Per i dettagli sull'API, vedere [RegisterOnPremisesInstance](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-CDApplication

Il seguente esempio di codice mostra come utilizzare. Remove-CDApplication

Strumenti per PowerShell

Esempio 1: questo esempio elimina l'applicazione con il nome specificato. Il comando richiederà una conferma prima di procedere. Aggiungi il parametro `-Force` per eliminare l'applicazione senza una richiesta.

```
Remove-CDApplication -ApplicationName MyNewApplication
```

- Per i dettagli sull'API, vedere [DeleteApplication](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-CDDeploymentConfig

Il seguente esempio di codice mostra come utilizzare `Remove-CDDeploymentConfig`

Strumenti per PowerShell

Esempio 1: questo esempio elimina la configurazione di distribuzione con il nome specificato. Il comando richiederà una conferma prima di procedere. Aggiungi il parametro `-Force` per eliminare la configurazione di distribuzione senza una richiesta.

```
Remove-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts
```

- Per i dettagli sull'API, vedere [DeleteDeploymentConfig](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-CDDeploymentGroup

Il seguente esempio di codice mostra come utilizzare `Remove-CDDeploymentGroup`

Strumenti per PowerShell

Esempio 1: questo esempio elimina il gruppo di distribuzione con il nome specificato per l'applicazione specificata. Il comando richiederà una conferma prima di procedere. Aggiungi il parametro `-Force` per eliminare il gruppo di distribuzione senza una richiesta.

```
Remove-CDDeploymentGroup -ApplicationName MyNewApplication -DeploymentGroupName  
MyNewDeploymentGroup
```

- Per i dettagli sull'API, vedere [DeleteDeploymentGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-CDOnPremiseInstanceTag

Il seguente esempio di codice mostra come utilizzare. Remove-CDOnPremiseInstanceTag

Strumenti per PowerShell

Esempio 1: Questo esempio elimina il tag specificato per l'istanza locale con il nome specificato. Il comando richiederà una conferma prima di procedere. Aggiungi il parametro -Force per eliminare il tag senza una richiesta.

```
Remove-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" =  
"Name"; "Value" = "CodeDeployDemo-OnPrem"}
```

- Per i dettagli sull'API, vedere [RemoveTagsFromOnPremisesInstances](#) in AWS Tools for PowerShell Cmdlet Reference.

Stop-CDDeployment

Il seguente esempio di codice mostra come utilizzare. Stop-CDDeployment

Strumenti per PowerShell

Esempio 1: questo esempio tenta di interrompere la distribuzione con l'ID di distribuzione specificato.

```
Stop-CDDeployment -DeploymentId d-LJQNREYEX
```

Output:

```
Status      StatusMessage  
-----  
Pending     Stopping Pending. Stopping to schedule commands in the deployment  
instances
```

- Per i dettagli sull'API, vedere [StopDeployment](#) in AWS Tools for PowerShell Cmdlet Reference.

Unregister-CDOnPremiseInstance

Il seguente esempio di codice mostra come utilizzare. Unregister-CDOnPremiseInstance

Strumenti per PowerShell

Esempio 1: questo esempio annulla la registrazione dell'istanza locale con il nome specificato.

```
Unregister-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

- Per i dettagli sull'API, vedere [DeregisterOnPremisesInstance](#) in Cmdlet Reference.AWS Tools for PowerShell

Update-CDApplication

Il seguente esempio di codice mostra come utilizzare. Update-CDApplication

Strumenti per PowerShell

Esempio 1: Questo esempio modifica il nome dell'applicazione specificata.

```
Update-CDApplication -ApplicationName MyNewApplication -NewApplicationName  
MyNewApplication-2
```

- Per i dettagli sull'API, vedere [UpdateApplication](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-CDDeploymentGroup

Il seguente esempio di codice mostra come utilizzare. Update-CDDeploymentGroup

Strumenti per PowerShell

Esempio 1: questo esempio modifica il nome del gruppo di distribuzione specificato per l'applicazione specificata.

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication -  
CurrentDeploymentGroupName MyNewDeploymentGroup -NewDeploymentGroupName  
MyNewDeploymentGroup-2
```

Esempio 2: Questo esempio mostra come specificare i gruppi di tag di istanza EC2 con cui un'istanza deve essere identificata per poter essere inclusa nell'ambiente sostitutivo di una distribuzione blu/verde.

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication -
CurrentDeploymentGroupName MyNewDeploymentGroup -NewDeploymentGroupName
MyNewDeploymentGroup-2 -Ec2TagSetList
@(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{Key=
```

- Per i dettagli sull'API, vedere [UpdateDeploymentGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

CodePipeline esempi di utilizzo di Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with CodePipeline.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Confirm-CPJob

Il seguente esempio di codice mostra come utilizzare `Confirm-CPJob`.

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene lo stato del lavoro specificato.

```
Confirm-CPJob -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE -Nonce 3
```

Output:

```
Value
-----
InProgress
```

- Per i dettagli sull'API, vedere [AcknowledgeJob](#) in AWS Tools for PowerShell Cmdlet Reference.

Disable-CPStageTransition

Il seguente esempio di codice mostra come utilizzare. `Disable-CPStageTransition`

Strumenti per PowerShell

Esempio 1: Questo esempio disabilita la transizione in entrata per lo stadio specificato nella pipeline specificata.

```
Disable-CPStageTransition -PipelineName CodePipelineDemo -Reason "Disabling temporarily." -StageName Beta -TransitionType Inbound
```

- Per i dettagli sull'API, vedere [DisableStageTransition](#) in AWS Tools for PowerShell Cmdlet Reference.

Enable-CPStageTransition

Il seguente esempio di codice mostra come utilizzare. `Enable-CPStageTransition`

Strumenti per PowerShell

Esempio 1: Questo esempio abilita la transizione in entrata per lo stadio specificato nella pipeline specificata.

```
Enable-CPStageTransition -PipelineName CodePipelineDemo -StageName Beta - TransitionType Inbound
```

- Per i dettagli sull'API, vedere [EnableStageTransition](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CPActionType

Il seguente esempio di codice mostra come utilizzare. `Get-CPActionType`

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene informazioni su tutte le azioni disponibili per il proprietario specificato.

```
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)
}
```

Output:

```
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}
```

- Per i dettagli sull'API, vedere [ListActionTypes](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CPActionableJobList

Il seguente esempio di codice mostra come utilizzare. `Get-CPActionableJobList`

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene informazioni su tutti i lavori eseguibili per la categoria di azione, il proprietario, il provider, la versione e i parametri di query specificati.

```
Get-CPActionableJobList -ActionTypeId_Category Build -ActionTypeId_Owner Custom
-ActionTypeId_Provider MyCustomProviderName -ActionTypeId_Version 1 -QueryParam
@{"ProjectName" = "MyProjectName"}
```

Output:

AccountId	Data	Id
----- -----	----	--
80398EXAMPLE f57a0EXAMPLE	Amazon.CodePipeline.Model.JobData 3	0de392f5-712d-4f41-ace3-

- Per i dettagli sull'API, vedere [PollForJobs](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CPJobDetail

Il seguente esempio di codice mostra come utilizzare. `Get-CPJobDetail`

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene informazioni generali sul lavoro specificato.

```
Get-CPJobDetail -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE
```

Output:

AccountId	Data	Id
-----	----	--

```
80398EXAMPLE    Amazon.CodePipeline.Model.JobData
f570dc12-5ef3-44bc-945a-6e133EXAMPLE
```

Esempio 2: Questo esempio ottiene informazioni dettagliate sul lavoro specificato.

```
$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)
```

Output:

```
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+1sAU=
InputArtifacts:
  MyApp
OutputArtifacts:
  MyAppBuild
PipelineContext:
  Name = Build
  PipelineName = CodePipelineDemo
  Stage = Build
```

- Per i dettagli sull'API, vedere [GetJobDetails](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CPPipeline

Il seguente esempio di codice mostra come utilizzare. Get-CPPipeline

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene informazioni generali sulla pipeline specificata.

```
Get-CPPipeline -Name CodePipelineDemo -Version 1
```

Output:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {Source, Build, Beta, TestStage}
Version       : 1
```

Esempio 2: Questo esempio ottiene informazioni dettagliate sulla tubazione specificata.

```
$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)
}
}

```

Output:

```

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

```

- Per i dettagli sull'API, vedere [GetPipeline](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CPPipelineList

Il seguente esempio di codice mostra come utilizzare. `Get-CPPipelineList`

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene un elenco di pipeline disponibili.

```
Get-CPPipelineList
```

Output:

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

- Per i dettagli sull'API, vedere [ListPipelines](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CPPipelineState

Il seguente esempio di codice mostra come utilizzare. `Get-CPPipelineState`

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene informazioni generali sugli stadi della pipeline specificata.

```
Get-CPPipelineState -Name CodePipelineDemo
```

Output:

```

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

```

Esempio 2: Questo esempio ottiene informazioni dettagliate sullo stato della pipeline specificata.

```

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)
}
}
```

Output:

```
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 =

```

- Per i dettagli sull'API, vedere [GetPipelineState](#) in AWS Tools for PowerShell Cmdlet Reference.

New-CPCustomActionType

Il seguente esempio di codice mostra come utilizzare `New-CPCustomActionType`

Strumenti per PowerShell

Esempio 1: Questo esempio crea una nuova azione personalizzata con le proprietà specificate.

```

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

```

Output:

```

ActionConfigurationProperties : {ProjectName}

```


Id	: Amazon.CodePipeline.Model.ActionTypeId
InputArtifactDetails	: Amazon.CodePipeline.Model.ArtifactDetails
OutputArtifactDetails	: Amazon.CodePipeline.Model.ArtifactDetails
Settings	: Amazon.CodePipeline.Model.ActionTypeSettings

- Per i dettagli sull'API, vedere [CreateCustomActionType](#) in AWS Tools for PowerShell Cmdlet Reference.

New-CPPipeline

Il seguente esempio di codice mostra come utilizzare. New-CPPipeline

Strumenti per PowerShell

Esempio 1: questo esempio crea una nuova pipeline con le impostazioni specificate.

```
$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
```

Output:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {Source, Beta}
Version       : 1
```

- Per i dettagli sull'API, vedere [CreatePipeline](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-CPCustomActionType

Il seguente esempio di codice mostra come utilizzare `Remove-CPCustomActionType`

Strumenti per PowerShell

Esempio 1: Questo esempio elimina l'azione personalizzata specificata. Il comando richiederà una conferma prima di procedere. Aggiungi il parametro `-Force` per eliminare l'azione personalizzata senza una richiesta.

```
Remove-CPCustomActionType -Category Build -Provider MyBuildProviderName -Version 1
```

- Per i dettagli sull'API, vedere [DeleteCustomActionType](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-CPPipeline

Il seguente esempio di codice mostra come utilizzare. Remove-CPPipeline

Strumenti per PowerShell

Esempio 1: questo esempio elimina la pipeline specificata. Il comando richiederà una conferma prima di procedere. Aggiungete il parametro -Force per eliminare la pipeline senza una richiesta.

```
Remove-CPPipeline -Name CodePipelineDemo
```

- Per i dettagli sull'API, vedere [DeletePipeline](#) in AWS Tools for PowerShell Cmdlet Reference.

Start-CPPipelineExecution

Il seguente esempio di codice mostra come utilizzare. Start-CPPipelineExecution

Strumenti per PowerShell

Esempio 1: Questo esempio inizia a eseguire la pipeline specificata.

```
Start-CPPipelineExecution -Name CodePipelineDemo
```

- Per i dettagli sull'API, vedere [StartPipelineExecution](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-CPPipeline

Il seguente esempio di codice mostra come utilizzare. Update-CPPipeline

Strumenti per PowerShell

Esempio 1: Questo esempio aggiorna la pipeline esistente specificata con le impostazioni specificate.

```
$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-CPipeline -Pipeline $pipeline
```

Output:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
```

```
Name      : CodePipelineDemo
RoleArn   : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages    : {InputFiles, TestDeployment}
Version   : 2
```

- Per i dettagli sull'API, vedere [UpdatePipeline](#) in AWS Tools for PowerShell Cmdlet Reference.

Esempi di identità di Amazon Cognito con Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon Cognito Identity. AWS Tools for PowerShell

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Get-CGIIIdentityPool

Il seguente esempio di codice mostra come utilizzare `Get-CGIIIdentityPool`.

Strumenti per PowerShell

Esempio 1: recupera le informazioni su uno specifico pool di identità in base al relativo id.

```
Get-CGIIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
```

Output:

```
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

```

- Per i dettagli sull'API, vedere [DescribeIdentityPool](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CGIIdentityPoolList

Il seguente esempio di codice mostra come utilizzare. `Get-CGIIdentityPoolList`

Strumenti per PowerShell

Esempio 1: recupera un elenco di pool di identità esistenti.

```
Get-CGIIdentityPoolList
```

Output:

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

- Per i dettagli sull'API, vedere [ListIdentityPools](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CGIIdentityPoolRole

Il seguente esempio di codice mostra come utilizzare. `Get-CGIIdentityPoolRole`

Strumenti per PowerShell

Esempio 1: ottiene le informazioni sui ruoli per uno specifico pool di identità.

```
Get-CGIIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [GetIdentityPoolRoles](#) in AWS Tools for PowerShell Cmdlet Reference.

New-CGIIIdentityPool

Il seguente esempio di codice mostra come utilizzare `New-CGIIIdentityPool`

Strumenti per PowerShell

Esempio 1: crea un nuovo pool di identità che consente identità non autenticate.

```
New-CGIIIdentityPool -AllowUnauthenticatedIdentities $true -IdentityPoolName
CommonTests13
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [CreateIdentityPool](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-CGIIIdentityPool

Il seguente esempio di codice mostra come utilizzare. Remove-CGIIIdentityPool

Strumenti per PowerShell

Esempio 1: elimina un pool di identità specifico.

```
Remove-CGIIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
```

- Per i dettagli sull'API, vedere [DeleteIdentityPool](#) in AWS Tools for PowerShell Cmdlet Reference.

Set-CGIIIdentityPoolRole

Il seguente esempio di codice mostra come utilizzare. Set-CGIIIdentityPoolRole

Strumenti per PowerShell

Esempio 1: configura lo specifico pool di identità per avere un ruolo IAM non autenticato.

```
Set-CGIIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1 -Role @{ "unauthenticated" = "arn:aws:iam::123456789012:role/CommonTests1Role" }
```

- Per i dettagli sull'API, vedere [SetIdentityPoolRoles](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-CGIIIdentityPool

Il seguente esempio di codice mostra come utilizzare. Update-CGIIIdentityPool

Strumenti per PowerShell

Esempio 1: aggiorna alcune proprietà del pool di identità, in questo caso il nome del pool di identità.


```
Update-CGIIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1 -IdentityPoolName NewPoolName
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [UpdateIdentityPool](#) in AWS Tools for PowerShell Cmdlet Reference.

AWS Config esempi di utilizzo di Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with AWS Config.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Add-CFGResourceTag

Il seguente esempio di codice mostra come utilizzare `Add-CFGResourceTag`.

Strumenti per PowerShell

Esempio 1: questo esempio associa il tag specificato alla risorsa ARN, che in questo caso è `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"}
```

- Per [TagResource](#) i dettagli AWS Tools for PowerShell sull'API, vedere in Cmdlet Reference.

Get-CFGAggregateComplianceByConfigRuleList

Il seguente esempio di codice mostra come utilizzare `Get-CFGAggregateComplianceByConfigRuleList`

Strumenti per PowerShell

Esempio 1: questo esempio recupera i dettagli dal filtro `ConfigurationAggregator` «kaju» per la regola di configurazione specificata ed espande/restituisce la «Conformità» della regola.

```
Get-CFGAggregateComplianceByConfigRuleList -ConfigurationAggregatorName kaju
-Filters_ConfigRuleName ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK | Select-Object -
ExpandProperty Compliance
```

Output:

```
ComplianceContributorCount      ComplianceType
-----
Amazon.ConfigService.Model.ComplianceContributorCount NON_COMPLIANT
```

Esempio 2: Questo esempio recupera i dettagli da quanto fornito `ConfigurationAggregator`, li filtra per l'account specificato per tutte le regioni incluse nell'aggregatore e restituisce ulteriormente la conformità di tutte le regole.

```
Get-CFGAggregateComplianceByConfigRuleList -ConfigurationAggregatorName
kaju -Filters_AccountId 123456789012 | Select-Object ConfigRuleName,
@{N="Compliance";E={$_.Compliance.ComplianceType}}
```

Output:

ConfigRuleName	Compliance
-----	-----
ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK	NON_COMPLIANT
ec2-instance-no-public-ip	NON_COMPLIANT
desired-instance-type	NON_COMPLIANT

- Per i dettagli sull'API, vedere [DescribeAggregateComplianceByConfigRules](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-CFGAggregateComplianceDetailsByConfigRule

Il seguente esempio di codice mostra come utilizzare. Get - CFGAggregateComplianceDetailsByConfigRule

Strumenti per PowerShell

Esempio 1: Questo esempio restituisce i risultati della valutazione selezionando l'output con resource-id e resource-type per la regola AWS Config " che si trovano nello stato 'COMPLIANTdesired-instance-type' per l'account, l'aggregatore, la regione e la regola di configurazione specificati

```
Get-CFGAggregateComplianceDetailsByConfigRule -AccountId 123456789012 -
AwsRegion eu-west-1 -ComplianceType COMPLIANT -ConfigRuleName desired-
instance-type -ConfigurationAggregatorName raju | Select-Object -
ExpandProperty EvaluationResultIdentifier | Select-Object -ExpandProperty
EvaluationResultQualifier
```

Output:

ConfigRuleName	ResourceId	ResourceType
-----	-----	-----
desired-instance-type	i-0f1bf2f34c5678d12	AWS::EC2::Instance
desired-instance-type	i-0fd12dd3456789123	AWS::EC2::Instance

- Per [GetAggregateComplianceDetailsByConfigRule](#) AWS Tools for PowerShell i dettagli sull'API, vedere in Cmdlet Reference.

Get-CFGAggregateConfigRuleComplianceSummary

Il seguente esempio di codice mostra come utilizzare. Get - CFGAggregateConfigRuleComplianceSummary

Strumenti per PowerShell

Esempio 1: questo esempio restituisce il numero di regole non conformi per l'aggregatore specificato.

```
(Get-CFGAggregateConfigRuleComplianceSummary -ConfigurationAggregatorName
raju).AggregateComplianceCounts.ComplianceSummary.NonCompliantResourceCount
```

Output:

```
CapExceeded CappedCount
-----
False      5
```

- Per i dettagli sull'API, vedere [GetAggregateConfigRuleComplianceSummary](#) in Cmdlet Reference. AWS Tools for PowerShell

Get-CFGAggregateDiscoveredResourceCount

Il seguente esempio di codice mostra come utilizzare. Get - CFGAggregateDiscoveredResourceCount

Strumenti per PowerShell

Esempio 1: Questo esempio restituisce il conteggio delle risorse per l'aggregatore dato filtrato per la regione us-east-1.

```
Get-CFGAggregateDiscoveredResourceCount -ConfigurationAggregatorName Master -
Filters_Region us-east-1
```

Output:

```

GroupByKey GroupedResourceCounts NextToken TotalDiscoveredResources
-----
{} 455

```

Esempio 2: questo esempio restituisce il conteggio delle risorse raggruppato per RESOURCE_TYPE per l'area filtrata per l'aggregatore specificato.

```

Get-CFGAggregateDiscoveredResourceCount -ConfigurationAggregatorName Master -
Filters_Region us-east-1 -GroupByKey RESOURCE_TYPE |
Select-Object -ExpandProperty GroupedResourceCounts

```

Output:

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

- Per i dettagli sull'API, vedere in Cmdlet Reference.
[GetAggregateDiscoveredResourceCounts](#) AWS Tools for PowerShell

Get-CFGAggregateDiscoveredResourceList

Il seguente esempio di codice mostra come utilizzare. Get-CFGAggregateDiscoveredResourceList

Strumenti per PowerShell

Esempio 1: Questo esempio restituisce gli identificatori di risorsa per il tipo di risorsa specificato aggregati nell'aggregatore «Irlanda». Per l'elenco dei tipi di risorse, consulta 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)
```

Output:

```
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
```

Esempio 2: Questo esempio restituisce il tipo di risorsa **AwsEC2SecurityGroup** denominato «default» per l'aggregatore specificato filtrato con la regione us-east-1.

```
Get-CFGAggregateDiscoveredResourceList -ConfigurationAggregatorName raju -
ResourceType ([Amazon.ConfigService.ResourceType]::AWSEC2SecurityGroup) -
Filters_Region us-east-1 -Filters_ResourceName default
```

Output:

```
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

```

- Per i dettagli sull'API, vedere [ListAggregateDiscoveredResources](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-CFGAggregateResourceConfig

Il seguente esempio di codice mostra come utilizzare. `Get-CFGAggregateResourceConfig`

Strumenti per PowerShell

Esempio 1: questo esempio restituisce l'elemento di configurazione per la risorsa data aggregata ed espande Configuration.

```

(Get-CFGAggregateResourceConfig -ResourceIdentifier_SourceRegion
us-east-1 -ResourceIdentifier_SourceAccountId 123456789012 -
ResourceIdentifier_ResourceId sg-4fc1d234 -ResourceIdentifier_ResourceType
([Amazon.ConfigService.ResourceType]::AWSEC2SecurityGroup) -
ConfigurationAggregatorName raju).Configuration | ConvertFrom-Json

```

Output:

```

{"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"}]
```

- Per i dettagli sull'API, vedere [GetAggregateResourceconfig-service](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CFGAggregateResourceConfigBatch

Il seguente esempio di codice mostra come utilizzare. Get-CFGAggregateResourceConfigBatch

Strumenti per PowerShell

Esempio 1: Questo esempio recupera l'elemento di configurazione corrente per la risorsa (identificata) presente nell'aggregatore specificato.

```
$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
```

Output:

```
BaseConfigurationItems  UnprocessedResourceIdentifiers
-----
{}                       {arn:aws:ec2:eu-west-1:123456789012:instance/
i-012e3cb4df567e8aa}
```

- Per i dettagli sull'API, vedere [BatchGetAggregateResourceconfig-service](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CFGAggregationAuthorizationList

Il seguente esempio di codice mostra come utilizzare. Get-CFGAggregationAuthorizationList

Strumenti per PowerShell

Esempio 1: questo esempio recupera le autorizzazioni concesse agli aggregatori.

```
Get-CFGAggregationAuthorizationList
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DescribeAggregationAuthorizations](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CFGComplianceByConfigRule

Il seguente esempio di codice mostra come utilizzare. Get-CFGComplianceByConfigRule

Strumenti per PowerShell

Esempio 1: questo esempio recupera i dettagli di conformità per la regola ebs-optimized-instance, per la quale non esistono risultati di valutazione correnti per la regola, quindi restituisce INSUFFICIENT_DATA

```
(Get-CFGComplianceByConfigRule -ConfigRuleName ebs-optimized-instance).Compliance
```

Output:

```
ComplianceContributorCount ComplianceType
-----
INSUFFICIENT_DATA
```

Esempio 2: questo esempio restituisce il numero di risorse non conformi per la regola ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK.

```
(Get-CFGComplianceByConfigRule -ConfigRuleName ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK -
ComplianceType NON_COMPLIANT).Compliance.ComplianceContributorCount
```

Output:

```
CapExceeded CappedCount
-----
False      2
```

- Per i [DescribeComplianceByConfigRule](#) dettagli AWS Tools for PowerShell sull'API, vedere in Cmdlet Reference.

Get-CFGComplianceByResource

Il seguente esempio di codice mostra come utilizzare. `Get-CFGComplianceByResource`

Strumenti per PowerShell

Esempio 1: questo esempio verifica il tipo di **AWS::SSM::ManagedInstanceInventory** risorsa per il tipo di conformità «COMPLIANT».

```
Get-CFGComplianceByResource -ComplianceType COMPLIANT -ResourceType
AWS::SSM::ManagedInstanceInventory
```

Output:

```
Compliance                ResourceId                ResourceType
-----
Amazon.ConfigService.Model.Compliance i-0123bcf4b567890e3
AWS::SSM::ManagedInstanceInventory
Amazon.ConfigService.Model.Compliance i-0a1234f6f5d6b78f7
AWS::SSM::ManagedInstanceInventory
```

- Per i dettagli sull'API, vedere [DescribeComplianceByResource](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CFGComplianceDetailsByConfigRule

Il seguente esempio di codice mostra come utilizzare. `Get-CFGComplianceDetailsByConfigRule`

Strumenti per PowerShell

Esempio 1: questo esempio ottiene i risultati della valutazione per la regola access-keys-rotated e restituisce l'output raggruppato per tipo di conformità

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-keys-rotated | Group-Object ComplianceType
```

Output:

```
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...
```

Esempio 2: questo esempio richiede i dettagli di conformità per la regola per le risorse COMPLIANT. access-keys-rotated

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-
keys-rotated -ComplianceType COMPLIANT | ForEach-Object
{$_ .EvaluationResultIdentifier.EvaluationResultQualifier}
```

Output:

```
ConfigRuleName      ResourceId          ResourceType
-----
access-keys-rotated BCAB1CDJ2LITAPVEW3JAH AWS::IAM::User
access-keys-rotated BCAB1CDJ2LITL3EHREM4Q AWS::IAM::User
```

- Per i dettagli sull'API, vedere [GetComplianceDetailsByConfigRule](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CFGComplianceDetailsByResource

Il seguente esempio di codice mostra come utilizzare. Get-CFGComplianceDetailsByResource

Strumenti per PowerShell

Esempio 1: Questo esempio di risultati di valutazione per la risorsa data.

```
Get-CFGComplianceDetailsByResource -ResourceId ABCD5STJ4EFGHIVEW6JAH -ResourceType
'AWS::IAM::User'
```

Output:

```
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          :
```

- Per i dettagli sull'API, vedere [GetComplianceDetailsByResource](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CFGComplianceSummaryByConfigRule

Il seguente esempio di codice mostra come utilizzare. Get - CFGComplianceSummaryByConfigRule

Strumenti per PowerShell

Esempio 1: questo esempio restituisce il numero di regole di Config non conformi.

```
Get-CFGComplianceSummaryByConfigRule -Select
ComplianceSummary.NonCompliantResourceCount
```

Output:

```
CapExceeded CappedCount
-----
False      9
```

- Per i dettagli sull'API, vedere [GetComplianceSummaryByConfigRule](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-CFGComplianceSummaryByResourceType

Il seguente esempio di codice mostra come utilizzare. Get-CFGComplianceSummaryByResourceType

Strumenti per PowerShell

Esempio 1: questo esempio restituisce il numero di risorse conformi o non conformi e converte l'output 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
  }
}
```

- Per i dettagli sull'API, vedere in Cmdlet Reference.
[GetComplianceSummaryByResourceType](#) AWS Tools for PowerShell

Get-CFGConfigRule

Il seguente esempio di codice mostra come utilizzare. Get-CFGConfigRule

Strumenti per PowerShell

Esempio 1: questo esempio elenca le regole di configurazione per l'account, con le proprietà selezionate.

```
Get-CFGConfigRule | Select-Object ConfigRuleName, ConfigRuleId, ConfigRuleArn,
  ConfigRuleState
```

Output:

```

ConfigRuleName                ConfigRuleId                ConfigRuleArn
-----                -
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

```

- Per i dettagli sull'API, vedere [DescribeConfigRules](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CFGConfigRuleEvaluationStatus

Il seguente esempio di codice mostra come utilizzare `Get-CFGConfigRuleEvaluationStatus`

Strumenti per PowerShell

Esempio 1: questo esempio restituisce le informazioni sullo stato delle regole di configurazione specificate.

```
Get-CFGConfigRuleEvaluationStatus -ConfigRuleName root-account-mfa-enabled, vpc-
flow-logs-enabled
```

Output:

```

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

```

- Per i dettagli sull'API, vedere [DescribeConfigRuleEvaluationStatus](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CFGConfigurationAggregatorList

Il seguente esempio di codice mostra come utilizzare `Get-CFGConfigurationAggregatorList`

Strumenti per PowerShell

Esempio 1: questo esempio restituisce tutti gli aggregatori per la regione/il conto.

```
Get-CFGConfigurationAggregatorList
```

Output:

```

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
```

- Per i dettagli sull'API, vedere [DescribeConfigurationAggregators](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CFGConfigurationAggregatorSourcesStatus

Il seguente esempio di codice mostra come utilizzare. Get-
CFGConfigurationAggregatorSourcesStatus

Strumenti per PowerShell

Esempio 1: questo esempio visualizza i campi richiesti per le fonti nell'aggregatore specificato.

```
Get-CFGConfigurationAggregatorSourcesStatus -ConfigurationAggregatorName raju |
  select SourceType, LastUpdateStatus, LastUpdateTime, SourceId
```

Output:

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

- Per i dettagli sull'API, vedere [DescribeConfigurationAggregatorSourcesStatus](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CFGConfigurationRecorder

Il seguente esempio di codice mostra come utilizzare. Get-CFGConfigurationRecorder

Strumenti per PowerShell

Esempio 1: Questo esempio restituisce i dettagli dei registratori di configurazione.


```
Get-CFGConfigurationRecorder | Format-List
```

Output:

```
Name           : default
RecordingGroup  : Amazon.ConfigService.Model.RecordingGroup
RoleARN        : arn:aws:iam::123456789012:role/aws-service-role/
config.amazonaws.com/AWSServiceRoleForConfig
```

- Per i dettagli sull'API, vedere [DescribeConfigurationRecorders](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CFGConfigurationRecorderStatus

Il seguente esempio di codice mostra come utilizzare. `Get-CFGConfigurationRecorderStatus`

Strumenti per PowerShell

Esempio 1: Questo esempio restituisce lo stato dei registratori di configurazione.

```
Get-CFGConfigurationRecorderStatus
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DescribeConfigurationRecorderStatus](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CFGConformancePack

Il seguente esempio di codice mostra come utilizzare. `Get-CFGConformancePack`

Strumenti per PowerShell

Esempio 1: questo esempio elenca tutti i pacchetti di conformità.

```
Get-CFGConformancePack
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DescribeConformancePacks](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CFGDeliveryChannel

Il seguente esempio di codice mostra come utilizzare. `Get-CFGDeliveryChannel`

Strumenti per PowerShell

Esempio 1: questo esempio recupera il canale di consegna per la regione e visualizza i dettagli.

```
Get-CFGDeliveryChannel -Region eu-west-1 | Select-Object Name, S3BucketName,
S3KeyPrefix,
@{N="DeliveryFrequency";E={$_.ConfigSnapshotDeliveryProperties.DeliveryFrequency}}
```

Output:

Name	S3BucketName	S3KeyPrefix	DeliveryFrequency
default	config-bucket-NA	my	TwentyFour_Hours

- Per i dettagli sull'API, vedere [DescribeDeliveryChannels](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-CFGResourceTag

Il seguente esempio di codice mostra come utilizzare. Get-CFGResourceTag

Strumenti per PowerShell

Esempio 1: questo esempio elenca i tag associati alla risorsa specificata

```
Get-CFGResourceTag -ResourceArn $rules[0].ConfigRuleArn
```

Output:

```
Key      Value
---      -
Version  1.3
```

- Per i dettagli sull'API, vedere [ListTagsForResource](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-CFGConformancePack

Il seguente esempio di codice mostra come utilizzare. Remove-CFGConformancePack

Strumenti per PowerShell

Esempio 1: questo esempio rimuove il pacchetto di conformità specificato, insieme a tutte le regole, le azioni correttive e i risultati della valutazione del pacchetto.

```
Remove-CFGConformancePack -ConformancePackName dono
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DeleteConformancePack](#) in AWS Tools for PowerShell Cmdlet Reference.

Write-CFGConformancePack

Il seguente esempio di codice mostra come utilizzare `Write-CFGConformancePack`

Strumenti per PowerShell

Esempio 1: Questo esempio crea un pacchetto di conformità, recuperando il modello dal file yml specificato.

```
Write-CFGConformancePack -ConformancePackName dono -DeliveryS3Bucket kt-ps-examples  
-TemplateBody (Get-Content C:\windows\temp\template.yml -Raw)
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [PutConformancePack](#) AWS Tools for PowerShell

Write-CFGDeliveryChannel

Il seguente esempio di codice mostra come utilizzare `Write-CFGDeliveryChannel`

Strumenti per PowerShell

Esempio 1: questo esempio modifica la proprietà `DeliveryFrequency` di un canale di consegna esistente.

```
Write-CFGDeliveryChannel -ConfigSnapshotDeliveryProperties_DeliveryFrequency  
TwentyFour_Hours -DeliveryChannelName default -DeliveryChannel_S3BucketName config-  
bucket-NA -DeliveryChannel_S3KeyPrefix my
```

- Per i dettagli sull'API, vedere [PutDeliveryChannel](#) in AWS Tools for PowerShell Cmdlet Reference.

Esempi di Device Farm con Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with Device Farm.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

New-DFUpload

Il seguente esempio di codice mostra come utilizzare `New-DFUpload`.

Strumenti per PowerShell

Esempio 1: questo esempio crea un caricamento AWS Device Farm per un'app Android. È possibile ottenere l'ARN del progetto dall'output di `New-DFProject` o `get-DFProjectList`. Utilizzate l'URL firmato nell'output `New-DFUpload` per caricare un file su Device Farm.

```
New-DFUpload -ContentType "application/octet-stream" -ProjectArn  
"arn:aws:devicefarm:us-west-2:123456789012:project:EXAMPLEa-7ec1-4741-9c1f-  
d3e04EXAMPLE" -Name "app.apk" -Type ANDROID_APP
```

- Per i dettagli sull'API, vedere [CreateUpload](#) in Cmdlet Reference. AWS Tools for PowerShell

AWS Directory Service esempi di utilizzo di Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with AWS Directory Service.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Add-DSIpRoute

Il seguente esempio di codice mostra come utilizzare Add-DSIpRoute.

Strumenti per PowerShell

Esempio 1: questo comando rimuove il tag di risorsa assegnato all'ID di directory specificato

```
Add-DSIpRoute -DirectoryId d-123456ijkl -IpRoute @{CidrIp ="203.0.113.5/32"} -UpdateSecurityGroupForDirectoryController $true
```

- Per i dettagli sull'API, vedere [AddIpRoutes](#) in AWS Tools for PowerShell Cmdlet Reference.

Add-DSResourceTag

Il seguente esempio di codice mostra come utilizzare Add-DSResourceTag

Strumenti per PowerShell

Esempio 1: questo comando aggiunge il Resource Tag all'ID di directory specificato

```
Add-DSResourceTag -ResourceId d-123456ijkl -Tag @{Key="myTag"; Value="mytgValue"}
```

- Per i dettagli sull'API, vedere [AddTagsToResource](#) in AWS Tools for PowerShell Cmdlet Reference.

Approve-DSTrust

Il seguente esempio di codice mostra come utilizzare Approve-DSTrust

Strumenti per PowerShell

Esempio 1: questo esempio chiama l'operazione dell' VerifyTrust API AWS Directory Service per il Trustid specificato.

```
Approve-DSTrust -TrustId t-9067157123
```

- Per i dettagli sull'API, vedere [VerifyTrust](#) in AWS Tools for PowerShell Cmdlet Reference.

Confirm-DSSharedDirectory

Il seguente esempio di codice mostra come utilizzare. Confirm-DSSharedDirectory

Strumenti per PowerShell

Esempio 1: questo esempio accetta una richiesta di condivisione della directory inviata dal proprietario della directory Account AWS.

```
Confirm-DSSharedDirectory -SharedDirectoryId d-9067012345
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [AcceptSharedDirectory](#) in AWS Tools for PowerShell Cmdlet Reference.

Connect-DSDirectory

Il seguente esempio di codice mostra come utilizzare. Connect-DSDirectory

Strumenti per PowerShell

Esempio 1: questo esempio crea un AD Connector per connettersi a una directory locale.

```
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
```

- Per i dettagli sull'API, vedere [ConnectDirectory](#) in AWS Tools for PowerShell Cmdlet Reference.

Deny-DSSharedDirectory

Il seguente esempio di codice mostra come utilizzare. Deny-DSSharedDirectory

Strumenti per PowerShell

Esempio 1: questo esempio rifiuta una richiesta di condivisione della directory inviata dall'account del proprietario della directory.

```
Deny-DSSharedDirectory -SharedDirectoryId d-9067012345
```

Output:

```
d-9067012345
```

- Per i dettagli sull'API, vedere [RejectSharedDirectory](#) in AWS Tools for PowerShell Cmdlet Reference.

Disable-DSDirectoryShare

Il seguente esempio di codice mostra come utilizzare. Disable-DSDirectoryShare

Strumenti per PowerShell

Esempio 1: Questo esempio interrompe la condivisione della directory tra il proprietario della directory e l'account consumatore.

```
Disable-DSDirectoryShare -DirectoryId d-123456ijkl -UnshareTarget_Id 123456784321 -
UnshareTarget_Type ACCOUNT
```


Output:

```
d-9067012345
```

- Per i dettagli sull'API, vedere [UnshareDirectory](#) in AWS Tools for PowerShell Cmdlet Reference.

Disable-DSLADAPS

Il seguente esempio di codice mostra come utilizzare. `Disable-DSLADAPS`

Strumenti per PowerShell

Esempio 1: questo esempio disattiva le chiamate sicure LDAP per la directory specificata.

```
Disable-DSLADAPS -DirectoryId d-123456ijkl -Type Client
```

- Per i dettagli sull'API, vedere [DisableLDAP](#) in Cmdlet Reference.AWS Tools for PowerShell

Disable-DSRadius

Il seguente esempio di codice mostra come utilizzare. `Disable-DSRadius`

Strumenti per PowerShell

Esempio 1: questo esempio disabilita il server RADIUS configurato per un AD Connector o una directory Microsoft AD.

```
Disable-DSRadius -DirectoryId d-123456ijkl
```

- Per i dettagli sull'API, vedere [DisableRadius](#) in AWS Tools for PowerShell Cmdlet Reference.

Disable-DSSso

Il seguente esempio di codice mostra come utilizzare. `Disable-DSSso`

Strumenti per PowerShell

Esempio 1: questo esempio disabilita il single sign-on per una directory.

```
Disable-DSSso -DirectoryId d-123456ijkl
```

- Per i dettagli sull'API, vedere [DisableSso](#) in AWS Tools for PowerShell Cmdlet Reference.

Enable-DSDirectoryShare

Il seguente esempio di codice mostra come utilizzare. `Enable-DSDirectoryShare`

Strumenti per PowerShell

Esempio 1: Questo esempio condivide una directory specificata nel tuo AWS account con un altro AWS account utilizzando il metodo Handshake.

```
Enable-DSDirectoryShare -DirectoryId d-123456ijkl -ShareTarget_Id 123456784321 -  
ShareMethod HANDSHAKE -ShareTarget_Type ACCOUNT
```

Output:

```
d-9067012345
```

- Per i dettagli sull'API, vedere [ShareDirectory](#) in AWS Tools for PowerShell Cmdlet Reference.

Enable-DSLdapS

Il seguente esempio di codice mostra come utilizzare. `Enable-DSLdapS`

Strumenti per PowerShell

Esempio 1: questo esempio attiva lo switch per la directory specifica in modo che utilizzi sempre le chiamate sicure LDAP.

```
Enable-DSLdapS -DirectoryId d-123456ijkl -Type Client
```

- Per i dettagli sull'API, vedere [EnableLDAPS in Cmdlet Reference](#).AWS Tools for PowerShell

Enable-DSRadius

Il seguente esempio di codice mostra come utilizzare. `Enable-DSRadius`

Strumenti per PowerShell

Esempio 1: Questo esempio abilita l'autenticazione a più fattori (MFA) con la configurazione del server RADIUS fornita per un AD Connector o una directory Microsoft AD.

```
Enable-DSRadius -DirectoryId d-123456ijkl  
-RadiusSettings_AuthenticationProtocol PAP  
-RadiusSettings_DisplayLabel Radius  
-RadiusSettings_RadiusPort 1812  
-RadiusSettings_RadiusRetry 4  
-RadiusSettings_RadiusServer 10.4.185.113  
-RadiusSettings_RadiusTimeout 50  
-RadiusSettings_SharedSecret wJalrXUtnFEMI
```

- Per i dettagli sull'API, vedere [EnableRadius](#) in AWS Tools for PowerShell Cmdlet Reference.

Enable-DSSso

Il seguente esempio di codice mostra come utilizzare. Enable-DSSso

Strumenti per PowerShell

Esempio 1: questo esempio abilita il single sign-on per una directory.

```
Enable-DSSso -DirectoryId d-123456ijkl
```

- Per i dettagli sull'API, vedere [EnableSso](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-DSCertificate

Il seguente esempio di codice mostra come utilizzare. Get-DSCertificate

Strumenti per PowerShell

Esempio 1: Questo esempio visualizza le informazioni sul certificato registrato per una connessione LDAP protetta.

```
Get-DSCertificate -DirectoryId d-123456ijkl -CertificateId c-906731e34f
```

Output:

```
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.
```

- Per i dettagli sull'API, vedere [DescribeCertificate](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-DSCertificateList

Il seguente esempio di codice mostra come utilizzare `Get-DSCertificateList`

Strumenti per PowerShell

Esempio 1: Questo esempio elenca tutti i certificati registrati per una connessione LDAP protetta per una directory specificata.

```
Get-DSCertificateList -DirectoryId d-123456ijkl
```

Output:

```
CertificateId CommonName          ExpiryDateTime      State
-----
c-906731e34f  contoso-EC2AMAZ-CTGG2NM-CA  4/15/2025 6:34:15 PM Registered
```

- Per i dettagli sull'API, vedere [ListCertificates](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-DSConditionalForwarder

Il seguente esempio di codice mostra come utilizzare `Get-DSConditionalForwarder`

Strumenti per PowerShell

Esempio 1: questo comando ottiene tutti i Conditional Forwarder configurati di un determinato ID di directory.

```
Get-DSConditionalForwarder -DirectoryId d-123456ijkl
```

Output:

```
DnsIpAddress      RemoteDomainName ReplicationScope
-----
{172.31.77.239} contoso.com      Domain
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [DescribeConditionalForwarders](#) AWS Tools for PowerShell

Get-DSDirectory

Il seguente esempio di codice mostra come utilizzare. `Get-DSDirectory`

Strumenti per PowerShell

Esempio 1: Questo comando ottiene informazioni sulle directory che appartengono a questo account.

```
Get-DSDirectory | Select-Object DirectoryId, Name, DnsIpAddress, Type
```

Output:

```
DirectoryId  Name                DnsIpAddress                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
```

- Per i dettagli sull'API, vedere [DescribeDirectories](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-DSDirectoryLimit

Il seguente esempio di codice mostra come utilizzare. `Get-DSDirectoryLimit`

Strumenti per PowerShell

Esempio 1: questo esempio visualizza le informazioni sui limiti delle directory per la regione us-east-1.

```
Get-DSDirectoryLimit -Region us-east-1
```

Output:

```
CloudOnlyDirectoriesCurrentCount : 1
CloudOnlyDirectoriesLimit         : 10
CloudOnlyDirectoriesLimitReached  : False
CloudOnlyMicrosoftADCurrentCount : 1
CloudOnlyMicrosoftADLimit        : 20
CloudOnlyMicrosoftADLimitReached : False
ConnectedDirectoriesCurrentCount  : 1
ConnectedDirectoriesLimit         : 10
```

- Per i dettagli sull'API, vedere [GetDirectoryLimits](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-DSDomainControllerList

Il seguente esempio di codice mostra come utilizzare. `Get-DSDomainControllerList`

Strumenti per PowerShell

Esempio 1: questo comando ottiene l'elenco dettagliato dei controller di dominio avviati per il suddetto `directory-id`

```
Get-DSDomainControllerList -DirectoryId d-123456ijkl
```

Output:

```
AvailabilityZone      : us-east-1b
DirectoryId          : d-123456ijkl
DnsIpAddress         : 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
DnsIpAddress    : 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
```

- Per i dettagli sull'API, vedere [DescribeDomainControllers](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-DSEventTopic

Il seguente esempio di codice mostra come utilizzare. Get-DSEventTopic

Strumenti per PowerShell

Esempio 1: questo comando mostra le informazioni dell'argomento SNS configurato per la notifica quando lo stato della directory cambia.

```
Get-DSEventTopic -DirectoryId d-123456ijkl
```

Output:

```
CreatedDateTime : 12/13/2019 11:15:32 AM
DirectoryId     : d-123456ijkl
Status         : Registered
TopicArn       : arn:aws:sns:us-east-1:123456781234:snstopicname
TopicName      : snstopicname
```

- Per i dettagli sull'API, vedere [DescribeEventTopics](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-DSIpRouteList

Il seguente esempio di codice mostra come utilizzare. Get-DSIpRouteList

Strumenti per PowerShell

Esempio 1: questo comando ottiene i blocchi di indirizzi IP pubblici configurati in Directory IP Routing

```
Get-DSIpRouteList -DirectoryId d-123456ijkl
```

Output:

```
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 :
```

- Per i dettagli sull'API, vedere [ListIpRoutes](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-DSLdapSetting

Il seguente esempio di codice mostra come utilizzare. Get-DSLdapSetting

Strumenti per PowerShell

Esempio 1: Questo esempio descrive lo stato della sicurezza LDAP per la directory specificata.

```
Get-DSLdapSetting -DirectoryId d-123456ijkl
```

Output:

```
LastUpdatedDateTime  LDAPSStatus  LDAPSStatusReason
-----
4/15/2020 6:51:03 PM Enabled      LDAPS is enabled successfully.
```

- Per i dettagli sull'API, vedere [DescribeLDAPSSettings](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-DSLogSubscriptionList

Il seguente esempio di codice mostra come utilizzare. Get-DSLogSubscriptionList

Strumenti per PowerShell

Esempio 1: questo comando ottiene le informazioni sulle sottoscrizioni di registro dell'id di directory specificato

```
Get-DSLogSubscriptionList -DirectoryId d-123456ijkl
```

Output:

```
DirectoryId  LogGroupName
SubscriptionCreatedDateTime
-----
-----
d-123456ijkl /aws/directoryservice/d-123456ijkl-lan2.example.com 12/14/2019 9:05:23
AM
```

- Per i dettagli sull'API, vedere [ListLogSubscriptions](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-DSResourceTag

Il seguente esempio di codice mostra come utilizzare. Get-DSResourceTag

Strumenti per PowerShell

Esempio 1: questo comando ottiene tutti i tag della directory specificata.

```
Get-DSResourceTag -ResourceId d-123456ijkl
```

Output:

```
Key  Value
---  -
myTag myTagValue
```

- Per i dettagli sull'API, vedere [ListTagsForResource](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-DSSchemaExtension

Il seguente esempio di codice mostra come utilizzare. `Get-DSSchemaExtension`

Strumenti per PowerShell

Esempio 1: questo esempio elenca tutte le estensioni dello schema applicate a una directory Microsoft AD.

```
Get-DSSchemaExtension -DirectoryId d-123456ijkl
```

Output:

```
Description           : ManagedADSchemaExtension
DirectoryId           : d-123456ijkl
EndTime               : 4/12/2020 10:30:49 AM
SchemaExtensionId     : e-9067306643
SchemaExtensionStatus : Completed
SchemaExtensionStatusReason : Schema updates are complete.
StartTime             : 4/12/2020 10:28:42 AM
```

- Per i dettagli sull'API, vedere [ListSchemaExtensions](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-DSSharedDirectory

Il seguente esempio di codice mostra come utilizzare. `Get-DSSharedDirectory`

Strumenti per PowerShell

Esempio 1: questo esempio ottiene le directory condivise del tuo account AWS

```
Get-DSSharedDirectory -OwnerDirectoryId d-123456ijkl -SharedDirectoryId d-9067012345
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DescribeSharedDirectories](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-DSSnapshot

Il seguente esempio di codice mostra come utilizzare. Get-DSSnapshot

Strumenti per PowerShell

Esempio 1: Questo comando ottiene informazioni sugli snapshot della directory specificata che appartengono a questo account.

```
Get-DSSnapshot -DirectoryId d-123456ijkl
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DescribeSnapshots](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-DSSnapshotLimit

Il seguente esempio di codice mostra come utilizzare. Get-DSSnapshotLimit

Strumenti per PowerShell

Esempio 1: questo comando ottiene i limiti delle istantanee manuali per una directory specificata.

```
Get-DSSnapshotLimit -DirectoryId d-123456ijkl
```

Output:

```
ManualSnapshotsCurrentCount ManualSnapshotsLimit ManualSnapshotsLimitReached
-----
0                            5                            False
```

- Per i dettagli sull'API, vedere [GetSnapshotLimits](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-DSTrust

Il seguente esempio di codice mostra come utilizzare. Get-DSTrust

Strumenti per PowerShell

Esempio 1: Questo comando ottiene le informazioni sulle relazioni di trust create per l'id di directory specificato.

```
Get-DSTrust -DirectoryId d-123456abcd
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DescribeTrusts](#) in AWS Tools for PowerShell Cmdlet Reference.

New-DSAlias

Il seguente esempio di codice mostra come utilizzare. `New-DSAlias`

Strumenti per PowerShell

Esempio 1: Questo comando crea un alias per una directory e assegna l'alias all'id di directory specificato.

```
New-DSAlias -DirectoryId d-123456ijkl -Alias MyOrgName
```

Output:

```
Alias      DirectoryId
-----
myorgname d-123456ijkl
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [CreateAlias](#) AWS Tools for PowerShell

New-DSComputer

Il seguente esempio di codice mostra come utilizzare. `New-DSComputer`

Strumenti per PowerShell

Esempio 1: Questo esempio crea un nuovo oggetto computer Active Directory.

```
New-DSComputer -DirectoryId d-123456ijkl -ComputerName ADMemberServer -Password $Password
```

Output:

```
ComputerAttributes      ComputerId
-----
{WindowsSamName, DistinguishedName} S-1-5-21-1191241402-978882507-2717148213-1662
ADMemberServer
```

- Per i dettagli sull'API, vedere [CreateComputer](#) in AWS Tools for PowerShell Cmdlet Reference.

New-DSConditionalForwarder

Il seguente esempio di codice mostra come utilizzare. `New-DSConditionalForwarder`

Strumenti per PowerShell

Esempio 1: Questo esempio crea un server d'inoltro condizionale nell'ID di directory specificato.
AWS

```
New-DSConditionalForwarder -DirectoryId d-123456ijkl -DnsIpAddress 172.31.36.96,172.31.10.56 -RemoteDomainName contoso.com
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [CreateConditionalForwarder](#) AWS Tools for PowerShell

New-DSDirectory

Il seguente esempio di codice mostra come utilizzare. `New-DSDirectory`

Strumenti per PowerShell

Esempio 1: questo esempio crea una nuova directory Simple AD.

```
New-DSDirectory -Name corp.example.com -Password $Password -Size Small -VpcSettings_VpcId vpc-123459d -VpcSettings_SubnetIds subnet-1234kkaa,subnet-5678ffbb
```

- Per i dettagli sull'API, vedere [CreateDirectory](#) in AWS Tools for PowerShell Cmdlet Reference.

New-DSLogSubscription

Il seguente esempio di codice mostra come utilizzare. `New-DSLogSubscription`

Strumenti per PowerShell

Esempio 1: questo esempio crea un abbonamento per inoltrare in tempo reale i log di sicurezza dei controller di dominio Directory Service al gruppo di CloudWatch log Amazon specificato nel tuo Account AWS.

```
New-DSLogSubscription -DirectoryId d-123456ijkl -LogGroupName /aws/directoryservice/d-123456ijkl-lan2.example.com
```

- Per i dettagli sull'API, consulta AWS Tools for PowerShell Cmdlet [CreateLogSubscription](#) Reference.

New-DSMicrosoftAD

Il seguente esempio di codice mostra come utilizzare. `New-DSMicrosoftAD`

Strumenti per PowerShell

Esempio 1: Questo esempio crea una nuova directory Microsoft AD in Cloud AWS.

```
New-DSMicrosoftAD -Name corp.example.com -Password $Password -edition Standard -VpcSettings_VpcId vpc-123459d -VpcSettings_SubnetIds subnet-1234kkaa,subnet-5678ffbb
```

- Per i dettagli sull'API, vedere [CreateMicrosoftAD](#) in AWS Tools for PowerShell Cmdlet Reference.

New-DSSnapshot

Il seguente esempio di codice mostra come utilizzare. `New-DSSnapshot`

Strumenti per PowerShell

Esempio 1: questo esempio crea un'istantanea della directory

```
New-DSSnapshot -DirectoryId d-123456ijkl
```

- Per i dettagli sull'API, vedere [CreateSnapshot](#) in AWS Tools for PowerShell Cmdlet Reference.

New-DSTrust

Il seguente esempio di codice mostra come utilizzare. `New-DSTrust`

Strumenti per PowerShell

Esempio 1: questo esempio crea un trust bidirezionale a livello di foresta tra la directory AWS Managed Microsoft AD e Microsoft Active Directory esistente in locale.

```
New-DSTrust -DirectoryId d-123456ijkl -RemoteDomainName contoso.com -TrustDirection  
Two-Way -TrustType Forest -TrustPassword $Password -ConditionalForwarderIpAddr  
172.31.36.96
```

Output:

```
t-9067157123
```

- Per i dettagli sull'API, vedere [CreateTrust](#) in AWS Tools for PowerShell Cmdlet Reference.

Register-DSCertificate

Il seguente esempio di codice mostra come utilizzare `Register-DSCertificate`

Strumenti per PowerShell

Esempio 1: Questo esempio registra un certificato per una connessione LDAP sicura.

```
$Certificate = Get-Content contoso.cer -Raw  
Register-DSCertificate -DirectoryId d-123456ijkl -CertificateData $Certificate
```

Output:

```
c-906731e350
```

- Per i dettagli sull'API, vedere [RegisterCertificate](#) in AWS Tools for PowerShell Cmdlet Reference.

Register-DSEventTopic

Il seguente esempio di codice mostra come utilizzare `Register-DSEventTopic`

Strumenti per PowerShell

Esempio 1: questo esempio associa una directory come editore a un argomento SNS.

```
Register-DSEventTopic -DirectoryId d-123456ijkl -TopicName snstopicname
```


- Per i dettagli sull'API, vedere [RegisterEventTopic](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-DSConditionalForwarder

Il seguente esempio di codice mostra come utilizzare. Remove-DSConditionalForwarder

Strumenti per PowerShell

Esempio 1: questo esempio rimuove il server d'inoltro condizionale che è stato impostato per Directory. AWS

```
Remove-DSConditionalForwarder -DirectoryId d-123456ijkl -RemoteDomainName  
contoso.com
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [DeleteConditionalForwarder](#) AWS Tools for PowerShell

Remove-DSDirectory

Il seguente esempio di codice mostra come utilizzare. Remove-DSDirectory

Strumenti per PowerShell

Esempio 1: Questo esempio elimina una AWS directory del servizio Directory (Simple AD/ Microsoft AD/AD Connector)

```
Remove-DSDirectory -DirectoryId d-123456ijkl
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [DeleteDirectory](#) AWS Tools for PowerShell

Remove-DSIpRoute

Il seguente esempio di codice mostra come utilizzare. Remove-DSIpRoute

Strumenti per PowerShell

Esempio 1: questo comando rimuove l'IP specificato dalle route IP configurate di Directory-ID.

```
Remove-DSIpRoute -DirectoryId d-123456ijkl -CidrIp 203.0.113.5/32
```

- Per i dettagli sull'API, vedere [RemovepRoutes](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-DSLogSubscription

Il seguente esempio di codice mostra come utilizzare. Remove-DSLogSubscription

Strumenti per PowerShell

Esempio 1: questo comando rimuove la sottoscrizione di registro dell'ID di directory specificato

```
Remove-DSLogSubscription -DirectoryId d-123456ijkl
```

- Per i dettagli sull'API, vedere [DeleteLogSubscription](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-DSResourceTag

Il seguente esempio di codice mostra come utilizzare. Remove-DSResourceTag

Strumenti per PowerShell

Esempio 1: questo comando rimuove il tag di risorsa assegnato all'ID di directory specificato

```
Remove-DSResourceTag -ResourceId d-123456ijkl -TagKey myTag
```

- Per i dettagli sull'API, vedere [RemoveTagsFromResource](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-DSSnapshot

Il seguente esempio di codice mostra come utilizzare. Remove-DSSnapshot

Strumenti per PowerShell

Esempio 1: questo esempio rimuove l'istantanea creata manualmente.

```
Remove-DSSnapshot -SnapshotId s-9068b488kc
```

- Per i dettagli sull'API, vedere [DeleteSnapshot](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-DSTrust

Il seguente esempio di codice mostra come utilizzare. Remove-DSTrust

Strumenti per PowerShell

Esempio 1: questo esempio rimuove la relazione di trust esistente tra AWS Managed AD Directory e un dominio esterno.

```
Get-DSTrust -DirectoryId d-123456ijkl -Select Trusts.TrustId | Remove-DSTrust
```

Output:

```
t-9067157123
```

- Per i dettagli sull'API, vedere [DeleteTrust](#) in AWS Tools for PowerShell Cmdlet Reference.

Reset-DSUserPassword

Il seguente esempio di codice mostra come utilizzare. Reset-DSUserPassword

Strumenti per PowerShell

Esempio 1: questo esempio reimposta la password dell'utente di Active Directory denominato ADUser in AWS Managed microsoft AD o Simple AD Directory

```
Reset-DSUserPassword -UserName ADUser -DirectoryId d-123456ijkl -NewPassword $Password
```

- Per i dettagli sull'API, vedere [ResetUserPassword](#) in Cmdlet Reference.AWS Tools for PowerShell

Restore-DSFromSnapshot

Il seguente esempio di codice mostra come utilizzare. Restore-DSFromSnapshot

Strumenti per PowerShell

Esempio 1: Questo esempio ripristina una directory utilizzando uno snapshot della directory esistente.

```
Restore-DSFromSnapshot -SnapshotId s-9068b488kc
```

- Per i dettagli sull'API, vedere [RestoreFromSnapshot](#) in Cmdlet Reference.AWS Tools for PowerShell

Set-DSDomainControllerCount

Il seguente esempio di codice mostra come utilizzare. Set-DSDomainControllerCount

Strumenti per PowerShell

Esempio 1: Questo esempio imposta il numero di controller di dominio su 3 per l'id di directory specificato.

```
Set-DSDomainControllerCount -DirectoryId d-123456ijkl -DesiredNumber 3
```

- Per i dettagli sull'API, vedere [UpdateNumberOfDomainControllers](#) in AWS Tools for PowerShell Cmdlet Reference.

Start-DSSchemaExtension

Il seguente esempio di codice mostra come utilizzare. Start-DSSchemaExtension

Strumenti per PowerShell

Esempio 1: questo esempio applica un'estensione dello schema a una directory di Microsoft AD.

```
$ldif = Get-Content D:\Users\Username\Downloads\ExtendedSchema.ldf -Raw
Start-DSSchemaExtension -DirectoryId d-123456ijkl -
CreateSnapshotBeforeSchemaExtension $true -Description ManagedADSchemaExtension -
LdifContent $ldif
```

Output:

```
e-9067306643
```

- Per i dettagli sull'API, vedere [StartSchemaExtension](#) in AWS Tools for PowerShell Cmdlet Reference.

Stop-DSSchemaExtension

Il seguente esempio di codice mostra come utilizzare. Stop-DSSchemaExtension

Strumenti per PowerShell

Esempio 1: questo esempio annulla un'estensione dello schema in corso a una directory di Microsoft AD.

```
Stop-DSSchemaExtension -DirectoryId d-123456ijkl -SchemaExtensionId e-9067306643
```

- Per i dettagli sull'API, vedere [CancelSchemaExtension](#) in AWS Tools for PowerShell Cmdlet Reference.

Unregister-DSCertificate

Il seguente esempio di codice mostra come utilizzare. Unregister-DSCertificate

Strumenti per PowerShell

Esempio 1: Questo esempio elimina dal sistema il certificato registrato per una connessione LDAP protetta.

```
Unregister-DSCertificate -DirectoryId d-123456ijkl -CertificateId c-906731e34f
```

- Per i dettagli sull'API, vedere [DeregisterCertificate](#) in AWS Tools for PowerShell Cmdlet Reference.

Unregister-DSEventTopic

Il seguente esempio di codice mostra come utilizzare. Unregister-DSEventTopic

Strumenti per PowerShell

Esempio 1: questo esempio rimuove la directory specificata come editore dell'argomento SNS specificato.

```
Unregister-DSEventTopic -DirectoryId d-123456ijkl -TopicName snstopicname
```

- Per i dettagli sull'API, vedere [DeregisterEventTopic](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-DSConditionalForwarder

Il seguente esempio di codice mostra come utilizzare. Update-DSConditionalForwarder

Strumenti per PowerShell

Esempio 1: questo esempio aggiorna un server d'inoltro condizionale che è stato impostato per la directory. AWS

```
Update-DSConditionalForwarder -DirectoryId d-123456ijkl -DnsIpAddress 172.31.36.96,172.31.16.108 -RemoteDomainName contoso.com
```

- Per i dettagli sull'API, vedere [UpdateConditionalForwarder](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-DSRadius

Il seguente esempio di codice mostra come utilizzare. Update-DSRadius

Strumenti per PowerShell

Esempio 1: Questo esempio aggiorna le informazioni del server RADIUS per un AD Connector o una directory Microsoft AD.

```
Update-DSRadius -DirectoryId d-123456ijkl -RadiusSettings_RadiusRetry 3
```

- Per i dettagli sull'API, vedere [UpdateRadius](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-DSTrust

Il seguente esempio di codice mostra come utilizzare. Update-DSTrust

Strumenti per PowerShell

Esempio 1: questo esempio aggiorna il SelectiveAuth parametro del trust-id specificato da Disabled a Enabled.

```
Update-DSTrust -TrustId t-9067157123 -SelectiveAuth Enabled
```

Output:

```
RequestId                                TrustId
-----                                -
138864a7-c9a8-4ad1-a828-eae479e85b45  t-9067157123
```

- Per i dettagli sull'API, vedere [UpdateTrust](#) in AWS Tools for PowerShell Cmdlet Reference.

AWS DMS esempi di utilizzo di Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with AWS DMS.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

New-DMSReplicationTask

Il seguente esempio di codice mostra come utilizzare `New-DMSReplicationTask`.

Strumenti per PowerShell

Esempio 1: questo esempio crea una nuova attività di replica del AWS Database Migration Service che utilizza `CdcStartTime` invece di `CdcStartPosition`. `MigrationType` è impostato

su "full-load-and-cdc«, il che significa che la tabella di destinazione deve essere vuota. La nuova attività è contrassegnata con un tag con una chiave Stage e un valore chiave Test. Per ulteriori informazioni sui valori utilizzati da questo cmdlet, vedere [Creating a Task \(https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Tasks.Creating.html\)](https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Tasks.Creating.html) nella AWS Database Migration Service User Guide.

```
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"
```

- Per i dettagli sull'API, vedere [CreateReplicationTask](#) in AWS Tools for PowerShell Cmdlet Reference.

Esempi di DynamoDB con Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell con DynamoDB.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Add-DDBIndexSchema

Il seguente esempio di codice mostra come utilizzare `Add-DDBIndexSchema`.

Strumenti per PowerShell

Esempio 1: crea un `TableSchema` oggetto vuoto e vi aggiunge una nuova definizione di indice secondario locale prima di scrivere l' `TableSchema` oggetto nella pipeline.

```
$schema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
"LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
$schema = New-DDBTableSchema
```

Output:

AttributeSchema	KeySchema
LocalSecondaryIndexSchema	
-----	-----

{LastPostDateTime}	{}
{LastPostIndex}	

Esempio 2: aggiunge una nuova definizione di indice secondario locale all' `TableSchema` oggetto fornito prima di riscrivere l' `TableSchema` oggetto nella pipeline. L' `TableSchema` oggetto può essere fornito anche utilizzando il parametro `-Schema`.

```
New-DDBTableSchema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
"LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
```

Output:

AttributeSchema	KeySchema
LocalSecondaryIndexSchema	
-----	-----

{LastPostDateTime}	{}
{LastPostIndex}	

- Per i dettagli sull'API, vedere [Add-DDB IndexSchema](#) in AWS Tools for PowerShell Cmdlet Reference.

Add-DDBKeySchema

Il seguente esempio di codice mostra come utilizzare. Add-DDBKeySchema

Strumenti per PowerShell

Esempio 1: crea un TableSchema oggetto vuoto e vi aggiunge le voci di definizione di chiavi e attributi utilizzando i dati chiave specificati prima di scrivere l' TableSchema oggetto nella pipeline. Il tipo di chiave è dichiarato «HASH» per impostazione predefinita; utilizzate il KeyType parametro - con il valore «RANGE» per dichiarare una chiave di intervallo.

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
```

Output:

AttributeSchema	KeySchema
LocalSecondaryIndexSchema	
-----	-----

{ForumName}	{ForumName}
{ }	

Esempio 2: aggiunge nuove voci di definizione di chiavi e attributi all' TableSchema oggetto fornito prima di scrivere l' TableSchema oggetto nella pipeline. Il tipo di chiave è dichiarato «HASH» per impostazione predefinita; utilizzate il KeyType parametro - con il valore «RANGE» per dichiarare una chiave di intervallo. L' TableSchema oggetto può essere fornito anche utilizzando il parametro -Schema.

```
New-DDBTableSchema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
```

Output:

AttributeSchema	KeySchema
LocalSecondaryIndexSchema	
-----	-----

```
{ForumName}
  {}
```

- Per i dettagli sull'API, vedere [Add-DDB KeySchema](#) in AWS Tools for PowerShell Cmdlet Reference.

ConvertFrom-DDBItem

Il seguente esempio di codice mostra come utilizzare. ConvertFrom-DDBItem

Strumenti per PowerShell

Esempio 1: ConvertFrom -DDBItem viene utilizzato per convertire il risultato di Get-DDBItem da una tabella hash di DynamoDB a una tabella hash di tipi comuni come string e AttributeValues double.

```
@{
  SongTitle = 'Somewhere Down The Road'
  Artist    = 'No One You Know'
} | ConvertTo-DDBItem

Get-DDBItem -TableName 'Music' -Key $key | ConvertFrom-DDBItem
```

Output:

Name	Value
----	-----
Genre	Country
Artist	No One You Know
Price	1.94
CriticRating	9
SongTitle	Somewhere Down The Road
AlbumTitle	Somewhat Famous

- [Per i dettagli sull'AWS Tools for PowerShell API, vedere -DDBItem in Cmdlet Reference. ConvertFrom](#)

ConvertTo-DDBItem

Il seguente esempio di codice mostra come utilizzare. ConvertTo-DDBItem

Strumenti per PowerShell

Esempio 1: Un esempio di conversione di una tabella hash in un dizionario di valori di attributi DynamoDB.

```
@{
    SongTitle = 'Somewhere Down The Road'
    Artist    = 'No One You Know'
} | ConvertTo-DDBItem
```

Key	Value
---	-----
SongTitle	Amazon.DynamoDBv2.Model.AttributeValue
Artist	Amazon.DynamoDBv2.Model.AttributeValue

Esempio 2: Un esempio per convertire una tabella hash in un dizionario di valori di attributi DynamoDB.

```
@{
    MyMap      = @{
        MyString = 'my string'
    }
    MyStringSet = [System.Collections.Generic.HashSet[String]]@('my', 'string')
    MyNumericSet = [System.Collections.Generic.HashSet[Int]]@(1, 2, 3)
    MyBinarySet = [System.Collections.Generic.HashSet[System.IO.MemoryStream]]@(
        ([IO.MemoryStream]::new([Text.Encoding]::UTF8.GetBytes('my'))),
        ([IO.MemoryStream]::new([Text.Encoding]::UTF8.GetBytes('string'))))
    MyList1     = @('my', 'string')
    MyList2     = [System.Collections.Generic.List[Int]]@(1, 2)
    MyList3     = [System.Collections.ArrayList]@('one', 2, $true)
} | ConvertTo-DDBItem
```

Output:

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

- Per i dettagli sull'API, vedere [ConvertTo-DbItem](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-DDBBatchItem

Il seguente esempio di codice mostra come utilizzare. Get-DDBBatchItem

Strumenti per PowerShell

Esempio 1: ottiene l'elemento con SongTitle «Somewhere Down The Road» dalle tabelle «Music» e «Songs» di DynamoDB.

```
$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
```

Output:

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

```

- Per i dettagli sull'API, vedere [BatchGetItem](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-DDBItem

Il seguente esempio di codice mostra come utilizzare. Get-DDBItem

Strumenti per PowerShell

Esempio 1: restituisce l'elemento DynamoDB con la chiave di partizione e la SongTitle chiave di ordinamento Artist.

```

$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem

Get-DDBItem -TableName 'Music' -Key $key | ConvertFrom-DDBItem

```

Output:

```

Name           Value
----           -
Genre          Country
SongTitle      Somewhere Down The Road
Price          1.94
Artist         No One You Know
CriticRating   9
AlbumTitle     Somewhat Famous

```

- Per i dettagli sull'API, vedere [GetItem](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-DDBTable

Il seguente esempio di codice mostra come utilizzare. `Get-DDBTable`

Strumenti per PowerShell

Esempio 1: restituisce i dettagli della tabella specificata.

```
Get-DDBTable -TableName "myTable"
```

- Per i dettagli sull'API, vedere [DescribeTable](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-DDBTableList

Il seguente esempio di codice mostra come utilizzare. `Get-DDBTableList`

Strumenti per PowerShell

Esempio 1: restituisce i dettagli di tutte le tabelle, iterando automaticamente fino a quando il servizio non indica che non esistono altre tabelle.

```
Get-DDBTableList
```

Esempio 2: esegue un'iterazione manuale per visualizzare i dettagli di tutte le tabelle, restituendo fino a 10 tabelle per chiamata finché il servizio non indica che non esistono altre tabelle.

```
$nextToken = $null
do {
    Get-DDBTableList -ExclusiveStartTableName $nextToken -Limit 10
    $nextToken = $AWSHistory.LastServiceResponse.LastEvaluatedTableName
} while ($nextToken -ne $null)
```

- Per i dettagli sull'API, vedere [ListTables](#) in AWS Tools for PowerShell Cmdlet Reference.

Invoke-DDBQuery

Il seguente esempio di codice mostra come utilizzare. `Invoke-DDBQuery`

Strumenti per PowerShell

Esempio 1: richiama una query che restituisce elementi DynamoDB con l'elemento Artist specificato. SongTitle

```
$invokeDDBQuery = @{
    TableName = 'Music'
    KeyConditionExpression = ' SongTitle = :SongTitle and Artist = :Artist'
    ExpressionAttributeValues = @{
        ':SongTitle' = 'Somewhere Down The Road'
        ':Artist' = 'No One You Know'
    } | ConvertTo-DDBItem
}
Invoke-DDBQuery @invokeDDBQuery | ConvertFrom-DDBItem
```

Output:

Name	Value
----	-----
Genre	Country
Artist	No One You Know
Price	1.94
CriticRating	9
SongTitle	Somewhere Down The Road
AlbumTitle	Somewhat Famous

- Per i dettagli sull'API, vedere [Query](#) in Cmdlet Reference.AWS Tools for PowerShell

Invoke-DDBScan

Il seguente esempio di codice mostra come utilizzare. Invoke-DDBScan

Strumenti per PowerShell

Esempio 1: restituisce tutti gli elementi della tabella Music.

```
Invoke-DDBScan -TableName 'Music' | ConvertFrom-DDBItem
```

Output:

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

```

Esempio 2: restituisce gli elementi nella tabella Music con un valore CriticRating maggiore o uguale a nove.

```

$scanFilter = @{
    CriticRating = [Amazon.DynamoDBv2.Model.Condition]{
        AttributeValueList = @( @{N = '9'} )
        ComparisonOperator = 'GE'
    }
}
Invoke-DDBScan -TableName 'Music' -ScanFilter $scanFilter | ConvertFrom-DDBItem

```

Output:

```

Name                Value
----                -
Genre                Country
Artist              No One You Know
Price               1.94
CriticRating        9
SongTitle           Somewhere Down The Road
AlbumTitle          Somewhat Famous

```

- Per i dettagli sull'API, vedere [Scan](#) in AWS Tools for PowerShell Cmdlet Reference.

New-DDBTable

Il seguente esempio di codice mostra come utilizzare. New-DDBTable

Strumenti per PowerShell

Esempio 1: Questo esempio crea una tabella denominata Thread con una chiave primaria composta da 'ForumName' (hash del tipo di chiave) e 'Subject' (intervallo dei tipi di chiave). Lo schema utilizzato per costruire la tabella può essere inserito in ogni cmdlet come mostrato o specificato utilizzando il parametro -Schema.

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
$schema | Add-DDBKeySchema -KeyName "Subject" -KeyType RANGE -KeyDataType "S"
$schema | New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

Output:

```
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 : {}
```

Esempio 2: questo esempio crea una tabella denominata Thread con una chiave primaria composta da " (hash del tipo di chiave) e ForumName 'Subject' (intervallo dei tipi di chiave). Viene inoltre definito un indice secondario locale. La chiave dell'indice secondario locale verrà impostata automaticamente dalla chiave hash primaria sulla tabella (ForumName). Lo schema utilizzato per costruire la tabella può essere inserito in ogni cmdlet come mostrato o specificato utilizzando il parametro -Schema.

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
$schema | Add-DDBKeySchema -KeyName "Subject" -KeyDataType "S"
$schema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
  "LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
$schema | New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

Output:

```
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}

```

Esempio 3: Questo esempio mostra come utilizzare una singola pipeline per creare una tabella denominata Thread con una chiave primaria composta da " (hash del tipo di chiave) e ForumName 'Subject' (intervallo dei tipi di chiave) e un indice secondario locale. Add-DB KeySchema e Add-DDB IndexSchema creano automaticamente un nuovo TableSchema oggetto se uno non viene fornito dalla pipeline o dal parametro -Schema.

```

New-DDBTableSchema |
  Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S" |
  Add-DDBKeySchema -KeyName "Subject" -KeyDataType "S" |
  Add-DDBIndexSchema -IndexName "LastPostIndex" `
    -RangeKeyName "LastPostDateTime" `
    -RangeKeyDataType "S" `
    -ProjectionType "keys_only" |
  New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5

```

Output:

```

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}

```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [CreateTable](#)AWS Tools for PowerShell

New-DDBTableSchema

Il seguente esempio di codice mostra come utilizzare. New-DDBTableSchema

Strumenti per PowerShell

Esempio 1: crea un TableSchema oggetto vuoto pronto ad accettare definizioni di chiavi e indici da utilizzare nella creazione di una nuova tabella Amazon DynamoDB. L'oggetto restituito può essere reindirizzato ai cmdlet Add-DDBKeySchema, Add-DDB IndexSchema e New-DDBTable o passato a tali cmdlet utilizzando il parametro -Schema su ciascun cmdlet.

```
New-DDBTableSchema
```

Output:

```
AttributeSchema                                KeySchema
  LocalSecondaryIndexSchema
-----
-----
{}                                               {}
  {}
```

- Per i [dettagli TableSchema AWS Tools for PowerShell](#) sull'API, vedere New-DDB in Cmdlet Reference.

Remove-DDBItem

Il seguente esempio di codice mostra come utilizzare. Remove-DDBItem

Strumenti per PowerShell

Esempio 1: rimuove l'elemento DynamoDB che corrisponde alla chiave fornita.

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem
Remove-DDBItem -TableName 'Music' -Key $key -Confirm:$false
```

- Per i dettagli sull'API, vedere [DeleteItem](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-DDBTable

Il seguente esempio di codice mostra come utilizzare. Remove-DDBTable

Strumenti per PowerShell

Esempio 1: elimina la tabella specificata. Prima di procedere con l'operazione, viene richiesta una conferma.

```
Remove-DDBTable -TableName "myTable"
```

Esempio 2: elimina la tabella specificata. Non viene richiesta la conferma prima che l'operazione proceda.

```
Remove-DDBTable -TableName "myTable" -Force
```

- Per i dettagli sull'API, vedere [DeleteTable](#) in AWS Tools for PowerShell Cmdlet Reference.

Set-DDBBatchItem

Il seguente esempio di codice mostra come utilizzare Set-DDBBatchItem

Strumenti per PowerShell

Esempio 1: crea un nuovo elemento o sostituisce un elemento esistente con un nuovo elemento nelle tabelle DynamoDB Music and 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
```

Output:

```
$requestItem = @{
    'Music' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
    'Songs' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
}
```

```
Set-DDBatchItem -RequestItem $requestItem
```

- Per i dettagli sull'API, vedere [BatchWriteItem](#) in AWS Tools for PowerShell Cmdlet Reference.

Set-DDBItem

Il seguente esempio di codice mostra come utilizzare. Set-DDBItem

Strumenti per PowerShell

Esempio 1: crea un nuovo elemento o sostituisce un elemento esistente con uno nuovo.

```
$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
```

- Per i dettagli sull'API, vedere [PutItem](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-DDBItem

Il seguente esempio di codice mostra come utilizzare. Update-DDBItem

Strumenti per PowerShell

Esempio 1: imposta l'attributo genere su 'Rap' sull'elemento DynamoDB con la chiave di partizione e la SongTitle chiave di ordinamento 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
```

Output:

Name	Value
----	-----
Genre	Rap

- Per i dettagli sull'API, vedere [UpdateItem](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-DDBTable

Il seguente esempio di codice mostra come utilizzare `Update-DDBTable`

Strumenti per PowerShell

Esempio 1: aggiorna il throughput assegnato per la tabella specificata.

```
Update-DDBTable -TableName "myTable" -ReadCapacity 10 -WriteCapacity 5
```

- Per i dettagli sull'API, vedere [UpdateTable](#) in AWS Tools for PowerShell Cmdlet Reference.

Esempi di Amazon EC2 che utilizzano Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon EC2. AWS Tools for PowerShell

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Add-EC2CapacityReservation

Il seguente esempio di codice mostra come utilizzare `Add-EC2CapacityReservation`.

Strumenti per PowerShell

Esempio 1: questo esempio crea una nuova riserva di capacità con gli attributi specificati

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

Output:

```
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
```

- Per i dettagli sull'API, vedere [CreateCapacityReservation](#) in AWS Tools for PowerShell Cmdlet Reference.

Add-EC2InternetGateway

Il seguente esempio di codice mostra come utilizzare `Add-EC2InternetGateway`

Strumenti per PowerShell

Esempio 1: questo esempio collega il gateway Internet specificato al VPC specificato.

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

Esempio 2: questo esempio crea un VPC e un gateway Internet, quindi collega il gateway Internet al VPC.

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

- Per i dettagli sull'API, vedere [AttachInternetGateway](#) in AWS Tools for PowerShell Cmdlet Reference.

Add-EC2NetworkInterface

Il seguente esempio di codice mostra come utilizzare Add-EC2NetworkInterface

Strumenti per PowerShell

Esempio 1: questo esempio collega l'interfaccia di rete specificata all'istanza specificata.

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

Output:

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

- Per i dettagli sull'API, vedere [AttachNetworkInterface](#) in AWS Tools for PowerShell Cmdlet Reference.

Add-EC2Volume

Il seguente esempio di codice mostra come utilizzare Add-EC2Volume

Strumenti per PowerShell

Esempio 1: Questo esempio collega il volume specificato all'istanza specificata e lo espone con il nome del dispositivo specificato.

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

Output:

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

- Per i dettagli sull'API, vedere [AttachVolume](#) in AWS Tools for PowerShell Cmdlet Reference.

Add-EC2VpnGateway

Il seguente esempio di codice mostra come utilizzare. Add-EC2VpnGateway

Strumenti per PowerShell

Esempio 1: questo esempio collega il gateway privato virtuale specificato al VPC specificato.

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

Output:

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

- Per i dettagli sull'API, vedere [AttachVpnGateway](#) in AWS Tools for PowerShell Cmdlet Reference.

Approve-EC2VpcPeeringConnection

Il seguente esempio di codice mostra come utilizzare. Approve-EC2VpcPeeringConnection

Strumenti per PowerShell

Esempio 1: Questo esempio approva il pcx-1dfad234b56ff78be richiesto VpcPeeringConnectionId

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

Output:

```
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
```

- Per i dettagli AWS Tools for PowerShell sull'[AcceptVpcPeeringConnection](#) API, vedere in Cmdlet Reference.

Confirm-EC2ProductInstance

Il seguente esempio di codice mostra come utilizzare. Confirm-EC2ProductInstance

Strumenti per PowerShell

Esempio 1: Questo esempio determina se il codice prodotto specificato è associato all'istanza specificata.

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

- Per i dettagli sull'API, vedere [ConfirmProductInstance](#) in AWS Tools for PowerShell Cmdlet Reference.

Copy-EC2Image

Il seguente esempio di codice mostra come utilizzare. Copy-EC2Image

Strumenti per PowerShell

Esempio 1: questo esempio copia l'AMI specificato nella regione «UE (Irlanda)» nella regione «Stati Uniti occidentali (Oregon)». Se -Region non è specificato, la regione predefinita corrente viene utilizzata come regione di destinazione.

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

Output:

```
ami-87654321
```

- Per i dettagli sull'API, vedere [CopyImage](#) in AWS Tools for PowerShell Cmdlet Reference.

Copy-EC2Snapshot

Il seguente esempio di codice mostra come utilizzare. Copy-EC2Snapshot

Strumenti per PowerShell

Esempio 1: Questo esempio copia l'istantanea specificata dalla regione UE (Irlanda) alla regione Stati Uniti occidentali (Oregon).

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

Esempio 2: se si imposta una regione predefinita e si omette il parametro Region, la regione di destinazione predefinita è la regione predefinita.

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

- Per i dettagli sull'API, vedere [CopySnapshot](#) in AWS Tools for PowerShell Cmdlet Reference.

Deny-EC2VpcPeeringConnection

Il seguente esempio di codice mostra come utilizzare. Deny-EC2VpcPeeringConnection

Strumenti per PowerShell

Esempio 1: L'esempio precedente nega la richiesta per l'id della richiesta VpcPeering pcx-01a2b3ce45fe67eb8

```
Deny-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-01a2b3ce45fe67eb8
```

- Per i dettagli [RejectVpcPeeringConnection](#) sull'AWS Tools for PowerShell API, vedere in Cmdlet Reference.

Disable-EC2VgwRoutePropagation

Il seguente esempio di codice mostra come utilizzare. `Disable-EC2VgwRoutePropagation`

Strumenti per PowerShell

Esempio 1: Questo esempio impedisce a VGW di propagare automaticamente le rotte alla tabella di routing specificata.

```
Disable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [DisableVgwRoutePropagation](#) AWS Tools for PowerShell

Disable-EC2VpcClassicLink

Il seguente esempio di codice mostra come utilizzare. `Disable-EC2VpcClassicLink`

Strumenti per PowerShell

Esempio 1: Questo esempio disabilita EC2 VpcClassicLink per vpc-01e23c4a5d6db78e9. Restituisce True o False

```
Disable-EC2VpcClassicLink -VpcId vpc-01e23c4a5d6db78e9
```

- Per i dettagli sull'API, vedere [DisableVpcClassicLink](#) in AWS Tools for PowerShell Cmdlet Reference.

Disable-EC2VpcClassicLinkDnsSupport

Il seguente esempio di codice mostra come utilizzare. `Disable-EC2VpcClassicLinkDnsSupport`

Strumenti per PowerShell

Esempio 1: Questo esempio disabilita il supporto ClassicLink DNS per vpc-0b12d3456a7e8910d

```
Disable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d
```

- Per i dettagli [DisableVpcClassicLinkDnsSupport](#) sull'API AWS Tools for PowerShell , vedere in Cmdlet Reference.

Dismount-EC2InternetGateway

Il seguente esempio di codice mostra come utilizzare. Dismount-EC2InternetGateway

Strumenti per PowerShell

Esempio 1: Questo esempio scollega il gateway Internet specificato dal VPC specificato.

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

- Per i dettagli sull'API, vedere [DetachInternetGateway](#) in AWS Tools for PowerShell Cmdlet Reference.

Dismount-EC2NetworkInterface

Il seguente esempio di codice mostra come utilizzare. Dismount-EC2NetworkInterface

Strumenti per PowerShell

Esempio 1: Questo esempio rimuove l'allegato specificato tra un'interfaccia di rete e un'istanza.

```
Dismount-EC2NetworkInterface -AttachmentId eni-attach-1a2b3c4d -Force
```

- Per i dettagli sull'API, vedere [DetachNetworkInterface](#) in AWS Tools for PowerShell Cmdlet Reference.

Dismount-EC2Volume

Il seguente esempio di codice mostra come utilizzare. Dismount-EC2Volume

Strumenti per PowerShell

Esempio 1: questo esempio rimuove il volume specificato.

```
Dismount-EC2Volume -VolumeId vol-12345678
```

Output:

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

Esempio 2: puoi anche specificare l'ID dell'istanza e il nome del dispositivo per assicurarti di scollegare il volume corretto.

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

- Per i dettagli sull'API, vedere [DetachVolume](#) in AWS Tools for PowerShell Cmdlet Reference.

Dismount-EC2VpnGateway

Il seguente esempio di codice mostra come utilizzare `Dismount-EC2VpnGateway`

Strumenti per PowerShell

Esempio 1: questo esempio scollega il gateway privato virtuale specificato dal VPC specificato.

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

- Per i dettagli sull'API, vedere [DetachVpnGateway](#) in AWS Tools for PowerShell Cmdlet Reference.

Edit-EC2CapacityReservation

Il seguente esempio di codice mostra come utilizzare `Edit-EC2CapacityReservation`

Strumenti per PowerShell

Esempio 1: questo esempio modifica `CapacityReservationId cr-0c1f2345db6f7cdba` cambiando il conteggio delle istanze su 1

```
Edit-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba -  
InstanceCount 1
```

Output:

```
True
```

- Per [ModifyCapacityReservation](#) i dettagli AWS Tools for PowerShell sull'API, vedere in Cmdlet Reference.

Edit-EC2Host

Il seguente esempio di codice mostra come utilizzare. `Edit-EC2Host`

Strumenti per PowerShell

Esempio 1: Questo esempio modifica le AutoPlacement impostazioni su off per l'host dedicato h-01e23f4cd567890f3

```
Edit-EC2Host -HostId h-03e09f8cd681609f3 -AutoPlacement off
```

Output:

```
Successful          Unsuccessful  
-----  
{h-01e23f4cd567890f3} {}
```

- Per [ModifyHosts](#) i dettagli sull'AWS Tools for PowerShell API, vedere in Cmdlet Reference.

Edit-EC2IdFormat

Il seguente esempio di codice mostra come utilizzare. `Edit-EC2IdFormat`

Strumenti per PowerShell

Esempio 1: Questo esempio abilita il formato ID più lungo per il tipo di risorsa specificato.

```
Edit-EC2IdFormat -Resource instance -UseLongId $true
```


Esempio 2: Questo esempio disabilita il formato ID più lungo per il tipo di risorsa specificato.

```
Edit-EC2IdFormat -Resource instance -UseLongId $false
```

- Per i dettagli sull'API, vedere [ModifyIdFormat](#) in AWS Tools for PowerShell Cmdlet Reference.

Edit-EC2ImageAttribute

Il seguente esempio di codice mostra come utilizzare `Edit-EC2ImageAttribute`

Strumenti per PowerShell

Esempio 1: questo esempio aggiorna la descrizione dell'AMI specificato.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Description "New description"
```

Esempio 2: questo esempio rende l'AMI pubblico (ad esempio, in modo che chiunque Account AWS possa utilizzarlo).

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType add -UserGroup all
```

Esempio 3: questo esempio rende l'AMI privato (ad esempio, in modo che solo tu come proprietario possa utilizzarlo).

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType remove -UserGroup all
```

Esempio 4: questo esempio concede il permesso di avvio all'oggetto specificato Account AWS.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType add -UserId 111122223333
```

Esempio 5: Questo esempio rimuove l'autorizzazione di avvio da quella specificata Account AWS.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType remove -UserId 111122223333
```

- Per i dettagli sull'API, vedere [ModifyImageAttribute](#) in AWS Tools for PowerShell Cmdlet Reference.

Edit-EC2InstanceAttribute

Il seguente esempio di codice mostra come utilizzare Edit-EC2InstanceAttribute

Strumenti per PowerShell

Esempio 1: Questo esempio modifica il tipo di istanza dell'istanza specificata.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceType m3.medium
```

Esempio 2: Questo esempio abilita una rete avanzata per l'istanza specificata, specificando «simple» come valore del parametro di supporto della rete di virtualizzazione I/O a radice singola (SR-IOV), -.. SriovNetSupport

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SriovNetSupport "simple"
```

Esempio 3: questo esempio modifica i gruppi di sicurezza per l'istanza specificata. L'istanza deve trovarsi in un VPC. È necessario specificare l'ID di ogni gruppo di sicurezza, non il nome.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -Group @( "sg-12345678",  
"sg-45678901" )
```

Esempio 4: questo esempio abilita l'ottimizzazione dell'I/O EBS per l'istanza specificata. Questa funzionalità non è disponibile con tutti i tipi di istanze. Quando si utilizza un'istanza ottimizzata per EBS, si applicano costi di utilizzo aggiuntivi.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -EbsOptimized $true
```

Esempio 5: questo esempio abilita il controllo di origine/destinazione per l'istanza specificata. Affinché un'istanza NAT esegua NAT, il valore deve essere 'false'.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SourceDestCheck $true
```

Esempio 6: questo esempio disabilita la terminazione per l'istanza specificata.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -DisableApiTermination $true
```

Esempio 7: Questo esempio modifica l'istanza specificata in modo che termini quando viene avviato lo spegnimento dall'istanza.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceInitiatedShutdownBehavior terminate
```

- Per i dettagli sull'API, vedere [ModifyInstanceAttribute](#) in Cmdlet Reference.AWS Tools for PowerShell

Edit-EC2InstanceCreditSpecification

Il seguente esempio di codice mostra come utilizzare. Edit-EC2InstanceCreditSpecification

Strumenti per PowerShell

Esempio 1: Ciò consente crediti illimitati per T2, ad esempio i-01234567890abcdef.

```
$Credit = New-Object -TypeName Amazon.EC2.Model.InstanceCreditSpecificationRequest  
$Credit.InstanceId = "i-01234567890abcdef"  
$Credit.CpuCredits = "unlimited"  
Edit-EC2InstanceCreditSpecification -InstanceCreditSpecification $Credit
```

- Per i dettagli [ModifyInstanceCreditSpecification](#) sull'AWS Tools for PowerShell API, vedere in Cmdlet Reference.

Edit-EC2NetworkInterfaceAttribute

Il seguente esempio di codice mostra come utilizzare. Edit-EC2NetworkInterfaceAttribute

Strumenti per PowerShell

Esempio 1: Questo esempio modifica l'interfaccia di rete specificata in modo che l'allegato specificato venga eliminato al termine.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -  
Attachment_AttachmentId eni-attach-1a2b3c4d -Attachment_DeleteOnTermination $true
```

Esempio 2: questo esempio modifica la descrizione dell'interfaccia di rete specificata.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Description "my description"
```

Esempio 3: Questo esempio modifica il gruppo di sicurezza per l'interfaccia di rete specificata.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Groups sg-1a2b3c4d
```

Esempio 4: Questo esempio disabilita il controllo di origine/destinazione per l'interfaccia di rete specificata.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -SourceDestCheck $false
```

- Per i dettagli sull'API, vedere [ModifyNetworkInterfaceAttribute](#) in Cmdlet Reference. AWS Tools for PowerShell

Edit-EC2ReservedInstance

Il seguente esempio di codice mostra come utilizzare. Edit-EC2ReservedInstance

Strumenti per PowerShell

Esempio 1: Questo esempio modifica la zona di disponibilità, il numero di istanze e la piattaforma per le istanze riservate specificate.

```
$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
```

- Per i dettagli sull'API, vedere [ModifyReservedInstances](#) in AWS Tools for PowerShell Cmdlet Reference.

Edit-EC2SnapshotAttribute

Il seguente esempio di codice mostra come utilizzare. Edit-EC2SnapshotAttribute

Strumenti per PowerShell

Esempio 1: questo esempio rende pubblica l'istantanea specificata impostandone l' CreateVolumePermission attributo.

```
Edit-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute  
CreateVolumePermission -OperationType Add -GroupName all
```

- Per i dettagli sull'API, vedere [ModifySnapshotAttribute](#) in AWS Tools for PowerShell Cmdlet Reference.

Edit-EC2SpotFleetRequest

Il seguente esempio di codice mostra come utilizzare. Edit-EC2SpotFleetRequest

Strumenti per PowerShell

Esempio 1: questo esempio aggiorna la capacità target della richiesta di flotta Spot specificata.

```
Edit-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-  
aa30-494c-8788-1cee4EXAMPLE -TargetCapacity 10
```

Output:

```
True
```

- Per i dettagli sull'API, vedere [ModifySpotFleetRequest](#) in AWS Tools for PowerShell Cmdlet Reference.

Edit-EC2SubnetAttribute

Il seguente esempio di codice mostra come utilizzare. Edit-EC2SubnetAttribute

Strumenti per PowerShell

Esempio 1: questo esempio abilita l'indirizzamento IP pubblico per la sottorete specificata.

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $true
```

Esempio 2: questo esempio disabilita l'indirizzamento IP pubblico per la sottorete specificata.

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $false
```

- Per i dettagli sull'API, vedere [ModifySubnetAttribute](#) in AWS Tools for PowerShell Cmdlet Reference.

Edit-EC2VolumeAttribute

Il seguente esempio di codice mostra come utilizzare Edit-EC2VolumeAttribute

Strumenti per PowerShell

Esempio 1: Questo esempio modifica l'attributo specificato del volume specificato. Le operazioni di I/O per il volume vengono riprese automaticamente dopo essere state sospese a causa di dati potenzialmente incoerenti.

```
Edit-EC2VolumeAttribute -VolumeId vol-12345678 -AutoEnableIO $true
```

- Per i dettagli sull'API, vedere [ModifyVolumeAttribute](#) in Cmdlet Reference.AWS Tools for PowerShell

Edit-EC2VpcAttribute

Il seguente esempio di codice mostra come utilizzare Edit-EC2VpcAttribute

Strumenti per PowerShell

Esempio 1: questo esempio abilita il supporto per i nomi host DNS per il VPC specificato.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $true
```

Esempio 2: questo esempio disabilita il supporto per i nomi host DNS per il VPC specificato.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $false
```

Esempio 3: questo esempio abilita il supporto per la risoluzione DNS per il VPC specificato.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $true
```

Esempio 4: questo esempio disabilita il supporto per la risoluzione DNS per il VPC specificato.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $false
```

- Per i dettagli sull'API, vedere [ModifyVpcAttribute](#) in AWS Tools for PowerShell Cmdlet Reference.

Enable-EC2VgwRoutePropagation

Il seguente esempio di codice mostra come utilizzare. `Enable-EC2VgwRoutePropagation`

Strumenti per PowerShell

Esempio 1: Questo esempio consente al VGW specificato di propagare automaticamente le rotte alla tabella di routing specificata.

```
Enable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- Per i dettagli sull'API, vedere [EnableVgwRoutePropagation](#) in Cmdlet Reference.AWS Tools for PowerShell

Enable-EC2VolumeIO

Il seguente esempio di codice mostra come utilizzare. `Enable-EC2VolumeIO`

Strumenti per PowerShell

Esempio 1: questo esempio abilita le operazioni di I/O per il volume specificato, se le operazioni di I/O sono state disabilitate.

```
Enable-EC2VolumeIO -VolumeId vol-12345678
```

- Per i dettagli sull'API, vedere [EnableVolumeIo](#) in AWS Tools for PowerShell Cmdlet Reference.

Enable-EC2VpcClassicLink

Il seguente esempio di codice mostra come utilizzare. `Enable-EC2VpcClassicLink`

Strumenti per PowerShell

Esempio 1: Questo esempio abilita VPC vpc-0123456b789b0d12f per ClassicLink

```
Enable-EC2VpcClassicLink -VpcId vpc-0123456b789b0d12f
```

Output:

```
True
```

- Per i dettagli sull'[EnableVpcClassicLink](#) API AWS Tools for PowerShell , vedere in Cmdlet Reference.

Enable-EC2VpcClassicLinkDnsSupport

Il seguente esempio di codice mostra come utilizzare. Enable-EC2VpcClassicLinkDnsSupport

Strumenti per PowerShell

Esempio 1: Questo esempio consente a vpc-0b12d3456a7e8910d di supportare la risoluzione dei nomi host DNS per ClassicLink

```
Enable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

- Per i dettagli [EnableVpcClassicLinkDnsSupport](#) sull'AWS Tools for PowerShell API, vedere in Cmdlet Reference.

Get-EC2AccountAttribute

Il seguente esempio di codice mostra come utilizzare. Get-EC2AccountAttribute

Strumenti per PowerShell

Esempio 1: Questo esempio descrive se è possibile avviare istanze in EC2-Classical ed EC2-VPC nella regione o solo in EC2-VPC.

```
(Get-EC2AccountAttribute -AttributeName supported-platforms).AttributeValues
```

Output:


```
AttributeValue
-----
EC2
VPC
```

Esempio 2: Questo esempio descrive il tuo VPC predefinito o è «nessuno» se non hai un VPC predefinito nella regione.

```
(Get-EC2AccountAttribute -AttributeName default-vpc).AttributeValues
```

Output:

```
AttributeValue
-----
vpc-12345678
```

Esempio 3: questo esempio descrive il numero massimo di istanze On-Demand che è possibile eseguire.

```
(Get-EC2AccountAttribute -AttributeName max-instances).AttributeValues
```

Output:

```
AttributeValue
-----
20
```

- Per i dettagli sull'API, vedere [DescribeAccountAttributes](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2Address

Il seguente esempio di codice mostra come utilizzare. `Get-EC2Address`

Strumenti per PowerShell

Esempio 1: Questo esempio descrive l'indirizzo IP elastico specificato per le istanze in EC2-Classical.

```
Get-EC2Address -AllocationId eipalloc-12345678
```

Output:

```
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
```

Esempio 2: questo esempio descrive gli indirizzi IP elastici per le istanze in un VPC. Questa sintassi richiede la PowerShell versione 3 o successiva.

```
Get-EC2Address -Filter @{ Name="domain";Values="vpc" }
```

Esempio 3: Questo esempio descrive l'indirizzo IP elastico specificato per le istanze in EC2-Classic.

```
Get-EC2Address -PublicIp 203.0.113.17
```

Output:

```
AllocationId      :
AssociationId     :
Domain           : standard
InstanceId       : i-12345678
NetworkInterfaceId :
NetworkInterfaceOwnerId :
PrivateIpAddress  :
PublicIp         : 203.0.113.17
```

Esempio 4: Questo esempio descrive gli indirizzi IP elastici per le istanze in EC2-Classic. Questa sintassi richiede la PowerShell versione 3 o successiva.

```
Get-EC2Address -Filter @{ Name="domain";Values="standard" }
```

Esempio 5: questo esempio descrive tutti i tuoi indirizzi IP elastici.

```
Get-EC2Address
```

Esempio 6: questo esempio restituisce l'IP pubblico e privato per l'ID dell'istanza fornito nel filtro

```
Get-EC2Address -Region eu-west-1 -Filter @{Name="instance-id";Values="i-0c12d3f4f567ffb89"} | Select-Object PrivateIpAddress, PublicIp
```

Output:

```
PrivateIpAddress PublicIp
-----
10.0.0.99          63.36.5.227
```

Esempio 7: questo esempio recupera tutti gli IP elastici con il relativo ID di allocazione, l'ID di associazione e gli ID di istanza

```
Get-EC2Address -Region eu-west-1 | Select-Object InstanceId, AssociationId, AllocationId, PublicIp
```

Output:

```
InstanceId          AssociationId        AllocationId        PublicIp
-----
17.212.120.178
i-0c123dfd3415bac67 eipassoc-0e123456bb7890bdb eipalloc-01cd23ebf45f7890c
17.212.124.77
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
```

Esempio 8: questo esempio recupera l'elenco di indirizzi IP EC2 che corrispondono alla chiave di tag «Category» con il valore «Prod»

```
Get-EC2Address -Filter @{"Name":"tag:Category";Values="Prod"}
```

Output:

```
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}
```

- Per i dettagli sull'API, vedere [DescribeAddresses](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2AvailabilityZone

Il seguente esempio di codice mostra come utilizzare. Get-EC2AvailabilityZone

Strumenti per PowerShell

Esempio 1: Questo esempio descrive le zone di disponibilità disponibili per la regione corrente.

```
Get-EC2AvailabilityZone
```

Output:

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
```

Esempio 2: questo esempio descrive tutte le zone di disponibilità che si trovano in uno stato compromesso. La sintassi utilizzata da questo esempio richiede la PowerShell versione 3 o successiva.

```
Get-EC2AvailabilityZone -Filter @{ Name="state";Values="impaired" }
```

Esempio 3: con PowerShell la versione 2, è necessario utilizzare New-Object per creare il filtro.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = "impaired"

Get-EC2AvailabilityZone -Filter $filter
```

- Per i dettagli sull'API, vedere [DescribeAvailabilityZones](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2BundleTask

Il seguente esempio di codice mostra come utilizzare. Get-EC2BundleTask

Strumenti per PowerShell

Esempio 1: questo esempio descrive l'attività di bundle specificata.

```
Get-EC2BundleTask -BundleId bun-12345678
```

Esempio 2: Questo esempio descrive le attività del pacchetto il cui stato è «completo» o «non riuscito».

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "complete", "failed" )

Get-EC2BundleTask -Filter $filter
```

- Per i dettagli sull'API, vedere [DescribeBundleTasks](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-EC2CapacityReservation

Il seguente esempio di codice mostra come utilizzare. `Get-EC2CapacityReservation`

Strumenti per PowerShell

Esempio 1: questo esempio descrive una o più delle tue prenotazioni di capacità per la regione

```
Get-EC2CapacityReservation -Region eu-west-1
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DescribeCapacityReservations](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2ConsoleOutput

Il seguente esempio di codice mostra come utilizzare. `Get-EC2ConsoleOutput`

Strumenti per PowerShell

Esempio 1: questo esempio ottiene l'output della console per l'istanza Linux specificata. L'output della console è codificato.

```
Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456
```

Output:

```

InstanceId      Output
-----
i-0e194d3c47c123637 WyAgICAwLjAwMDAwMF0gQ29tbW...bGU9dHR5UzAgc2Vs

```

Esempio 2: questo esempio memorizza l'output codificato della console in una variabile e quindi lo decodifica.

```

$output_encoded = (Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456).Output
[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($output_encoded))

```

- Per i dettagli sull'API, vedere [GetConsoleOutput](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2CustomerGateway

Il seguente esempio di codice mostra come utilizzare `Get-EC2CustomerGateway`

Strumenti per PowerShell

Esempio 1: questo esempio descrive il customer gateway specificato.

```
Get-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

Output:

```

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

```

Esempio 2: questo esempio descrive qualsiasi gateway per i clienti il cui stato è in sospeso o disponibile.

```

$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"

```

```
$filter.Values = @( "pending", "available" )

Get-EC2CustomerGateway -Filter $filter
```

Esempio 3: questo esempio descrive tutti i gateway per i clienti.

```
Get-EC2CustomerGateway
```

- Per i dettagli sull'API, vedere [DescribeCustomerGateways](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2DhcpOption

Il seguente esempio di codice mostra come utilizzare. Get-EC2DhcpOption

Strumenti per PowerShell

Esempio 1: Questo esempio elenca i set di opzioni DHCP.

```
Get-EC2DhcpOption
```

Output:

DhcpConfigurations	DhcpOptionsId	Tag
{domain-name, domain-name-servers}	dopt-1a2b3c4d	{}
{domain-name, domain-name-servers}	dopt-2a3b4c5d	{}
{domain-name-servers}	dopt-3a4b5c6d	{}

Esempio 2: Questo esempio ottiene i dettagli di configurazione per il set di opzioni DHCP specificato.

```
(Get-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d).DhcpConfigurations
```

Output:

Key	Values
---	-----


```
domain-name           {abc.local}
domain-name-servers   {10.0.0.101, 10.0.0.102}
```

- Per i dettagli sull'API, vedere [DescribeDhcpOptions](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2FlowLog

Il seguente esempio di codice mostra come utilizzare. Get-EC2FlowLog

Strumenti per PowerShell

Esempio 1: questo esempio descrive uno o più log di flusso con il tipo di destinazione dei log 's3'

```
Get-EC2FlowLog -Filter @{"Name="log-destination-type";Values="s3"}
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DescribeFlowLogs](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2Host

Il seguente esempio di codice mostra come utilizzare. Get-EC2Host

Strumenti per PowerShell

Esempio 1: questo esempio restituisce i dettagli dell'host EC2

```
Get-EC2Host
```

Output:

```
AllocationTime      : 3/23/2019 4:55:22 PM
AutoPlacement       : off
AvailabilityZone     : eu-west-1b
AvailableCapacity   : Amazon.EC2.Model.AvailableCapacity
ClientToken         :
HostId              : h-01e23f4cd567899f1
HostProperties       : Amazon.EC2.Model.HostProperties
HostReservationId   :
Instances           : {}
ReleaseTime         : 1/1/0001 12:00:00 AM
State               : available
Tags                : {}
```

Esempio 2: questo esempio esegue una query `AvailableInstanceCapacity` per l'host `h-01e23f4cd567899f1`

```
Get-EC2Host -HostId h-01e23f4cd567899f1 | Select-Object -ExpandProperty
AvailableCapacity | Select-Object -expand AvailableInstanceCapacity
```

Output:

```
AvailableCapacity InstanceType TotalCapacity
-----
11                m4.xlarge    11
```

- Per i dettagli [DescribeHosts](#) AWS Tools for PowerShell sull'API, vedere in Cmdlet Reference.

Get-EC2HostReservationOffering

Il seguente esempio di codice mostra come utilizzare `Get-EC2HostReservationOffering`

Strumenti per PowerShell

Esempio 1: questo esempio descrive le prenotazioni di host dedicati disponibili per l'acquisto con il filtro specificato «instance-family» where `PaymentOption` is «» `NoUpfront`

```
Get-EC2HostReservationOffering -Filter @{Name="instance-family";Values="m4"} |  
Where-Object PaymentOption -eq NoUpfront
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DescribeHostReservationOfferings](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2HostReservationPurchasePreview

Il seguente esempio di codice mostra come utilizzare. `Get-EC2HostReservationPurchasePreview`

Strumenti per PowerShell

Esempio 1: Questo esempio visualizza in anteprima l'acquisto di una prenotazione con configurazioni che corrispondono a quelle dell'host dedicato h-01e23f4cd567890f1

```
Get-EC2HostReservationPurchasePreview -OfferingId hro-0c1f23456789d0ab -HostIdSet  
h-01e23f4cd567890f1
```

Output:

```
CurrencyCode Purchase TotalHourlyPrice TotalUpfrontPrice
```

```
-----
      {}      1.307      0.000
```

- Per i [GetHostReservationPurchasePreview](#) dettagli sull'AWS Tools for PowerShell API, vedere in Cmdlet Reference.

Get-EC2IdFormat

Il seguente esempio di codice mostra come utilizzare. `Get-EC2IdFormat`

Strumenti per PowerShell

Esempio 1: Questo esempio descrive il formato ID per il tipo di risorsa specificato.

```
Get-EC2IdFormat -Resource instance
```

Output:

```
Resource      UseLongIds
-----
instance      False
```

Esempio 2: Questo esempio descrive i formati ID per tutti i tipi di risorse che supportano ID più lunghi.

```
Get-EC2IdFormat
```

Output:

```
Resource      UseLongIds
-----
reservation   False
instance      False
```

- Per i dettagli sull'API, vedere [DescribeIdFormat](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2IdentityIdFormat

Il seguente esempio di codice mostra come utilizzare. `Get-EC2IdentityIdFormat`

Strumenti per PowerShell

Esempio 1: questo esempio restituisce il formato ID per l'immagine della risorsa per il ruolo assegnato

```
Get-EC2IdentityIdFormat -PrincipalArn arn:aws:iam::123456789511:role/JDBC -Resource
image
```

Output:

```
Deadline          Resource UseLongIds
-----          -
8/2/2018 11:30:00 PM image      True
```

- Per i dettagli sull'API, vedere [DescribeIdentityIdFormat](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2Image

Il seguente esempio di codice mostra come utilizzare. Get-EC2Image

Strumenti per PowerShell

Esempio 1: questo esempio descrive l'AMI specificato.

```
Get-EC2Image -ImageId ami-12345678
```

Output:

```
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
```

Esempio 2: questo esempio descrive le AMI che possiedi.

```
Get-EC2Image -owner self
```

Esempio 3: questo esempio descrive le AMI pubbliche che eseguono Microsoft Windows Server.

```
Get-EC2Image -Filter @{ Name="platform"; Values="windows" }
```

Esempio 4: questo esempio descrive tutte le AMI pubbliche nella regione 'us-west-2'.

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

- Per i dettagli sull'API, vedere [DescribeImages](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2ImageAttribute

Il seguente esempio di codice mostra come utilizzare `Get-EC2ImageAttribute`

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene la descrizione dell'AMI specificato.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute description
```

Output:

```
BlockDeviceMappings : {}
```

```
Description      : My image description
ImageId          : ami-12345678
KernelId        :
LaunchPermissions : {}
ProductCodes    : {}
RamdiskId       :
SriovNetSupport  :
```

Esempio 2: Questo esempio ottiene i permessi di avvio per l'AMI specificato.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

Output:

```
BlockDeviceMappings : {}
Description          :
ImageId              : ami-12345678
KernelId             :
LaunchPermissions    : {all}
ProductCodes        : {}
RamdiskId            :
SriovNetSupport      :
```

Esempio 3: Questo esempio verifica se la rete avanzata è abilitata.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute sriovNetSupport
```

Output:

```
BlockDeviceMappings : {}
Description          :
ImageId              : ami-12345678
KernelId             :
LaunchPermissions    : {}
ProductCodes        : {}
RamdiskId            :
SriovNetSupport      : simple
```

- Per i dettagli sull'API, vedere [DescribeImageAttribute](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2ImageByName

Il seguente esempio di codice mostra come utilizzare. Get-EC2ImageByName

Strumenti per PowerShell

Esempio 1: Questo esempio descrive il set completo di nomi di filtri attualmente supportati.

```
Get-EC2ImageByName
```

Output:

```
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
```

Esempio 2: questo esempio descrive l'AMI specificato. L'utilizzo di questo comando per individuare un'AMI è utile perché AWS rilascia nuove AMI Windows con gli ultimi aggiornamenti ogni mese. È possibile specificare 'ImageId' per New-EC2Instance avviare un'istanza utilizzando l'AMI corrente per il filtro specificato.

```
Get-EC2ImageByName -Names WINDOWS_2016_BASE
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [Get-EC2ImageByName](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2ImportImageTask

Il seguente esempio di codice mostra come utilizzare `Get-EC2ImportImageTask`

Strumenti per PowerShell

Esempio 1: Questo esempio descrive l'attività di importazione dell'immagine specificata.

```
Get-EC2ImportImageTask -ImportTaskId import-ami-hgfedcba
```

Output:

```
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     :
```

Esempio 2: questo esempio descrive tutte le attività di importazione delle immagini.

```
Get-EC2ImportImageTask
```

Output:

```
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   :

```

- Per i dettagli sull'API, vedere [DescribeImportImageTasks](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2ImportSnapshotTask

Il seguente esempio di codice mostra come utilizzare. Get-EC2ImportSnapshotTask

Strumenti per PowerShell

Esempio 1: Questo esempio descrive l'attività di importazione delle istantanee specificata.

```
Get-EC2ImportSnapshotTask -ImportTaskId import-snap-abcdefgh
```

Output:

Description	ImportTaskId	SnapshotTaskDetail
-----	-----	-----
Disk Image Import 1	import-snap-abcdefgh	Amazon.EC2.Model.SnapshotTaskDetail

Esempio 2: questo esempio descrive tutte le attività di importazione delle istantanee.

```
Get-EC2ImportSnapshotTask
```

Output:

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

- Per i dettagli sull'API, vedere [DescribeImportSnapshotTasks](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2Instance

Il seguente esempio di codice mostra come utilizzare. Get-EC2Instance

Strumenti per PowerShell

Esempio 1: questo esempio descrive l'istanza specificata.

```
(Get-EC2Instance -InstanceId i-12345678).Instances
```

Output:

```
AmiLaunchIndex      : 0
Architecture        : x86_64
BlockDeviceMappings : {/dev/sda1}
ClientToken          : T1eEy1448154045270
EbsOptimized         : False
Hypervisor           : xen
IamInstanceProfile   : Amazon.EC2.Model.IamInstanceProfile
ImageId              : ami-12345678
InstanceId           : i-12345678
InstanceLifecycle    :
InstanceType         : t2.micro
KernelId             :
KeyName              : my-key-pair
LaunchTime           : 12/4/2015 4:44:40 PM
Monitoring           : Amazon.EC2.Model.Monitoring
NetworkInterfaces    : {ip-10-0-2-172.us-west-2.compute.internal}
```

```

Placement      : Amazon.EC2.Model.Placement
Platform       : Windows
PrivateDnsName : ip-10-0-2-172.us-west-2.compute.internal
PrivateIpAddress : 10.0.2.172
ProductCodes   : {}
PublicDnsName  :
PublicIpAddress :
RamdiskId      :
RootDeviceName : /dev/sda1
RootDeviceType : ebs
SecurityGroups : {default}
SourceDestCheck : True
SpotInstanceRequestId :
SriovNetSupport :
State          : Amazon.EC2.Model.InstanceState
StateReason    :
StateTransitionReason :
SubnetId       : subnet-12345678
Tags           : {Name}
VirtualizationType : hvm
VpcId          : vpc-12345678

```

Esempio 2: questo esempio descrive tutte le istanze nella regione corrente, raggruppate per prenotazione. Per visualizzare i dettagli dell'istanza, espandi la raccolta Instances all'interno di ciascun oggetto di prenotazione.

```
Get-EC2Instance
```

Output:

```

GroupNames    : {}
Groups        : {}
Instances     : {}
OwnerId       : 123456789012
RequesterId   : 226008221399
ReservationId : r-c5df370c

GroupNames    : {}
Groups        : {}
Instances     : {}
OwnerId       : 123456789012
RequesterId   : 854251627541

```

```
ReservationId : r-63e65bab
...
```

Esempio 3: Questo esempio illustra l'utilizzo di un filtro per interrogare le istanze EC2 in una sottorete specifica di un VPC.

```
(Get-EC2Instance -Filter @{Name="vpc-id";Values="vpc-1a2bc34d"},@{Name="subnet-id";Values="subnet-1a2b3c4d"}).Instances
```

Output:

```
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
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [DescribeInstances](#) AWS Tools for PowerShell

Get-EC2InstanceAttribute

Il seguente esempio di codice mostra come utilizzare `Get-EC2InstanceAttribute`

Strumenti per PowerShell

Esempio 1: Questo esempio descrive il tipo di istanza dell'istanza specificata.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute instanceType
```

Output:

```
InstanceType           : t2.micro
```

Esempio 2: Questo esempio descrive se la rete avanzata è abilitata per l'istanza specificata.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

Output:

```
SriovNetSupport           : simple
```

Esempio 3: Questo esempio descrive i gruppi di sicurezza per l'istanza specificata.

```
(Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute groupSet).Groups
```

Output:

```
GroupId
-----
sg-12345678
sg-45678901
```

Esempio 4: questo esempio descrive se l'ottimizzazione EBS è abilitata per l'istanza specificata.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

Output:

```
EbsOptimized           : False
```

Esempio 5: questo esempio descrive l'attributo disableApiTermination " dell'istanza specificata.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

Output:

```
DisableApiTermination   : False
```

Esempio 6: Questo esempio descrive l'attributo 'instanceInitiatedShutdownComportamento' dell'istanza specificata.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
instanceInitiatedShutdownBehavior
```

Output:

```
InstanceInitiatedShutdownBehavior : stop
```

- Per i dettagli sull'API, vedere [DescribeInstanceAttribute](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2InstanceMetadata

Il seguente esempio di codice mostra come utilizzare. Get-EC2InstanceMetadata

Strumenti per PowerShell

Esempio 1: elenca le categorie disponibili di metadati delle istanze su cui è possibile eseguire query.

```
Get-EC2InstanceMetadata -ListCategory
```

Output:

```
AmiId
LaunchIndex
ManifestPath
AncestorAmiId
BlockDeviceMapping
InstanceId
InstanceType
LocalHostname
LocalIpv4
KernelId
AvailabilityZone
ProductCode
PublicHostname
PublicIpv4
PublicKey
RamdiskId
Region
ReservationId
SecurityGroup
UserData
InstanceMonitoring
IdentityDocument
IdentitySignature
```



```
IdentityPkcs7
```

Esempio 2: restituisce l'id dell'Amazon Machine Image (AMI) utilizzato per avviare l'istanza.

```
Get-EC2InstanceMetadata -Category AmiId
```

Output:

```
ami-b2e756ca
```

Esempio 3: questo esempio interroga il documento di identità in formato JSON per l'istanza.

```
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"
}
```

Esempio 4: questo esempio utilizza una query di percorso per ottenere i mac dell'interfaccia di rete per l'istanza.

```
Get-EC2InstanceMetadata -Path "/network/interfaces/macs"
```

Output:

```
02:80:7f:ef:4c:e0/
```

Esempio 5: se all'istanza è associato un ruolo IAM, restituisce informazioni sull'ultima volta in cui il profilo dell'istanza è stato aggiornato, inclusa la LastUpdated data dell'istanza InstanceProfileArn, e InstanceProfileId.

```
Get-EC2InstanceMetadata -Path "/iam/info"
```

Output:

```
{
  "Code" : "Success",
  "LastUpdated" : "2018-03-08T03:38:40Z",
  "InstanceProfileArn" : "arn:aws:iam::111122223333:instance-profile/
MyLaunchRole_Profile",
  "InstanceProfileId" : "AIPAI4...WVK2RW"
}
```

- Per i dettagli sull'API, vedere [Get-EC2InstanceMetadata](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2InstanceStatus

Il seguente esempio di codice mostra come utilizzare. Get-EC2InstanceStatus

Strumenti per PowerShell

Esempio 1: questo esempio descrive lo stato dell'istanza specificata.

```
Get-EC2InstanceStatus -InstanceId i-12345678
```

Output:

```
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
```

Output:

```
Code      Name
----      -
16        running
```

```
$status.Status
```

Output:

```
Details          Status
-----          -
{reachability}  ok
```

```
$status.SystemStatus
```

Output:

```
Details          Status
-----          -
{reachability}  ok
```

- Per i dettagli sull'API, vedere [DescribeInstanceStatus](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2InternetGateway

Il seguente esempio di codice mostra come utilizzare. Get-EC2InternetGateway

Strumenti per PowerShell

Esempio 1: Questo esempio descrive il gateway Internet specificato.

```
Get-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

Output:

Attachments	InternetGatewayId	Tags
-----	-----	----
{vpc-1a2b3c4d}	igw-1a2b3c4d	{}

Esempio 2: questo esempio descrive tutti i gateway Internet.

```
Get-EC2InternetGateway
```

Output:

Attachments	InternetGatewayId	Tags
-----	-----	----
{vpc-1a2b3c4d}	igw-1a2b3c4d	{}
{}	igw-2a3b4c5d	{}

- Per i dettagli sull'API, vedere [DescribeInternetGateways](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2KeyPair

Il seguente esempio di codice mostra come utilizzare `Get-EC2KeyPair`

Strumenti per PowerShell

Esempio 1: Questo esempio descrive la coppia di key pair specificata.

```
Get-EC2KeyPair -KeyName my-key-pair
```

Output:

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

Esempio 2: questo esempio descrive tutte le coppie di chiavi.

```
Get-EC2KeyPair
```

- Per i dettagli sull'API, vedere [DescribeKeyPairs](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2NetworkAcl

Il seguente esempio di codice mostra come utilizzare `Get-EC2NetworkAcl`

Strumenti per PowerShell

Esempio 1: questo esempio descrive l'ACL di rete specificato.

```
Get-EC2NetworkAcl -NetworkAclId acl-12345678
```

Output:

```
Associations : {aclassoc-1a2b3c4d}
Entries      : {Amazon.EC2.Model.NetworkAclEntry, Amazon.EC2.Model.NetworkAclEntry}
IsDefault   : False
NetworkAclId : acl-12345678
Tags        : {Name}
VpcId       : vpc-12345678
```

Esempio 2: questo esempio descrive le regole per l'ACL di rete specificato.

```
(Get-EC2NetworkAcl -NetworkAclId acl-12345678).Entries
```

Output:

```
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
```

Esempio 3: questo esempio descrive tutti gli ACL di rete.

```
Get-EC2NetworkACL
```

- Per i dettagli sull'API, vedere [DescribeNetworkAcls](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2NetworkInterface

Il seguente esempio di codice mostra come utilizzare `Get-EC2NetworkInterface`

Strumenti per PowerShell

Esempio 1: Questo esempio descrive l'interfaccia di rete specificata.

```
Get-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

Output:

```
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
```

Esempio 2: questo esempio descrive tutte le interfacce di rete.

```
Get-EC2NetworkInterface
```

- Per i dettagli sull'API, vedere [DescribeNetworkInterfaces](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2NetworkInterfaceAttribute

Il seguente esempio di codice mostra come utilizzare `Get-EC2NetworkInterfaceAttribute`

Strumenti per PowerShell

Esempio 1: Questo esempio descrive l'interfaccia di rete specificata.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute Attachment
```

Output:

```
Attachment          : Amazon.EC2.Model.NetworkInterfaceAttachment
```

Esempio 2: questo esempio descrive l'interfaccia di rete specificata.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute Description
```

Output:

```
Description         : My description
```

Esempio 3: questo esempio descrive l'interfaccia di rete specificata.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute GroupSet
```

Output:

```
Groups              : {my-security-group}
```

Esempio 4: Questo esempio descrive l'interfaccia di rete specificata.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute SourceDestCheck
```

Output:

```
SourceDestCheck      : True
```

- Per i dettagli sull'API, vedere [DescribeNetworkInterfaceAttribute](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2PasswordData

Il seguente esempio di codice mostra come utilizzare. Get-EC2PasswordData

Strumenti per PowerShell

Esempio 1: questo esempio decrittografa la password che Amazon EC2 ha assegnato all'account Administrator per l'istanza Windows specificata. Poiché è stato specificato un file pem, viene automaticamente assunta l'impostazione dello switch -Decrypt.

```
Get-EC2PasswordData -InstanceId i-12345678 -PemFile C:\path\my-key-pair.pem
```

Output:

```
mYZ(PA9?C)Q
```

Esempio 2: (PowerShell solo Windows) Ispeziona l'istanza per determinare il nome della coppia di chiavi utilizzata per avviare l'istanza, quindi tenta di trovare i dati della coppia di chiavi corrispondente nell'archivio di configurazione di AWS Toolkit for Visual Studio. Se i dati della coppia di chiavi vengono trovati, la password viene decrittografata.

```
Get-EC2PasswordData -InstanceId i-12345678 -Decrypt
```

Output:

```
mYZ(PA9?C)Q
```

Esempio 3: restituisce i dati della password crittografata per l'istanza.


```
Get-EC2PasswordData -InstanceId i-12345678
```

Output:

```
iVz3BAK/WAXV.....dqt8WeMA==
```

- Per i dettagli sull'API, vedere [GetPasswordData](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2PlacementGroup

Il seguente esempio di codice mostra come utilizzare. `Get-EC2PlacementGroup`

Strumenti per PowerShell

Esempio 1: Questo esempio descrive il gruppo di posizionamento specificato.

```
Get-EC2PlacementGroup -GroupName my-placement-group
```

Output:

GroupName	State	Strategy
-----	-----	-----
my-placement-group	available	cluster

- Per i dettagli sull'API, vedere [DescribePlacementGroups](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2PrefixList

Il seguente esempio di codice mostra come utilizzare. `Get-EC2PrefixList`

Strumenti per PowerShell

Esempio 1: Questo esempio Servizi AWS recupera il formato di elenco di prefissi disponibile per la regione

```
Get-EC2PrefixList
```

Output:

Cidrs	PrefixListId	PrefixListName
-----	-----	-----
{52.94.5.0/24, 52.119.240.0/21, 52.94.24.0/23}	p1-6fa54006	com.amazonaws.eu-west-1.dynamodb
{52.218.0.0/17, 54.231.128.0/19}	p1-6da54004	com.amazonaws.eu-west-1.s3

- Per i dettagli sull'API, vedere [DescribePrefixLists](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2Region

Il seguente esempio di codice mostra come utilizzare. `Get-EC2Region`

Strumenti per PowerShell

Esempio 1: questo esempio descrive le aree a tua disposizione.

```
Get-EC2Region
```

Output:

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

- Per i dettagli sull'API, vedere [DescribeRegions](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2RouteTable

Il seguente esempio di codice mostra come utilizzare. `Get-EC2RouteTable`

Strumenti per PowerShell

Esempio 1: questo esempio descrive tutte le tabelle dei percorsi.

```
Get-EC2RouteTable
```

Output:

```
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 :
```

Esempio 2: questo esempio restituisce i dettagli per la tabella di percorso specificata.

```
Get-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

Esempio 3: questo esempio descrive le tabelle di routing per il VPC specificato.

```
Get-EC2RouteTable -Filter @{ Name="vpc-id"; Values="vpc-1a2b3c4d" }
```

Output:

```
Associations    : {rtbassoc-12345678}
PropagatingVgws : {}
Routes         : {, }
```

```
RouteTableId    : rtb-1a2b3c4d
Tags            : {}
VpcId          : vpc-1a2b3c4d
```

- Per i dettagli sull'API, vedere [DescribeRouteTables](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2ScheduledInstance

Il seguente esempio di codice mostra come utilizzare. Get-EC2ScheduledInstance

Strumenti per PowerShell

Esempio 1: questo esempio descrive l'istanza pianificata specificata.

```
Get-EC2ScheduledInstance -ScheduledInstanceId sci-1234-1234-1234-1234-123456789012
```

Output:

```
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
```

Esempio 2: questo esempio descrive tutte le istanze pianificate.

```
Get-EC2ScheduledInstance
```

- Per i dettagli sull'API, vedere [DescribeScheduledInstances](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2ScheduledInstanceAvailability

Il seguente esempio di codice mostra come utilizzare. Get-EC2ScheduledInstanceAvailability

Strumenti per PowerShell

Esempio 1: Questo esempio descrive una pianificazione che si verifica ogni settimana di domenica, a partire dalla data specificata.

```
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
```

Output:

```
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
...
```

Esempio 2: per restringere i risultati, è possibile aggiungere filtri per criteri quali sistema operativo, rete e tipo di istanza.

```
-Filter @{ Name="platform";Values="Linux/UNIX" },@{ Name="network-
platform";Values="EC2-VPC" },@{ Name="instance-type";Values="c4.large" }
```

- Per i dettagli sull'API, vedere [DescribeScheduledInstanceAvailability](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2SecurityGroup

Il seguente esempio di codice mostra come utilizzare `Get-EC2SecurityGroup`

Strumenti per PowerShell

Esempio 1: questo esempio descrive il gruppo di sicurezza specificato per un VPC. Quando si lavora con gruppi di sicurezza appartenenti a un VPC, è necessario utilizzare l'ID del gruppo di sicurezza (`GroupId` parametro -), non il nome (`GroupName` parametro -), per fare riferimento al gruppo.

```
Get-EC2SecurityGroup -GroupId sg-12345678
```

Output:

```
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
```

Esempio 2: Questo esempio descrive il gruppo di sicurezza specificato per EC2-Classic. Quando si lavora con gruppi di sicurezza per EC2-Classic, è possibile utilizzare il nome del gruppo (`GroupName` parametro) o l'ID del gruppo (`GroupId` parametro -) per fare riferimento al gruppo di sicurezza.

```
Get-EC2SecurityGroup -GroupName my-security-group
```

Output:

```
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 :
```

Esempio 3: Questo esempio recupera tutti i gruppi di sicurezza per vpc-0fc1ff23456b789eb

```
Get-EC2SecurityGroup -Filter @{"Name"="vpc-id";"Values"="vpc-0fc1ff23456b789eb"}
```

- Per AWS Tools for PowerShell i dettagli [DescribeSecurityGroups](#) sull'API, vedere in Cmdlet Reference.

Get-EC2Snapshot

Il seguente esempio di codice mostra come utilizzare. Get-EC2Snapshot

Strumenti per PowerShell

Esempio 1: questo esempio descrive l'istantanea specificata.

```
Get-EC2Snapshot -SnapshotId snap-12345678
```

Output:

```
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
```

Esempio 2: Questo esempio descrive le istantanee che hanno un tag 'Name'.

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" }
```

Esempio 3: Questo esempio descrive le istantanee che hanno un tag 'Nome' con il valore "TestValue"

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" -and
  $_.Tags.Value -eq "TestValue" }
```

Esempio 4: questo esempio descrive tutte le istantanee.

```
Get-EC2Snapshot -Owner self
```

- Per i dettagli sull'API, vedere [DescribeSnapshots](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2SnapshotAttribute

Il seguente esempio di codice mostra come utilizzare Get-EC2SnapshotAttribute

Strumenti per PowerShell

Esempio 1: Questo esempio descrive l'attributo specificato dell'istanza specificata.

```
Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute ProductCodes
```

Output:

```
CreateVolumePermissions    ProductCodes    SnapshotId
-----
{}                          {}              snap-12345678
```

Esempio 2: Questo esempio descrive l'attributo specificato dell'istanza specificata.

```
(Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
  CreateVolumePermission).CreateVolumePermissions
```

Output:

```
Group    UserId
-----
```



```
all
```

- Per i dettagli sull'API, vedere [DescribeSnapshotAttribute](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2SpotDatafeedSubscription

Il seguente esempio di codice mostra come utilizzare. `Get-EC2SpotDatafeedSubscription`

Strumenti per PowerShell

Esempio 1: questo esempio descrive il feed di dati dell'istanza Spot.

```
Get-EC2SpotDatafeedSubscription
```

Output:

```
Bucket   : my-s3-bucket
Fault    :
OwnerId  : 123456789012
Prefix   : spotdata
State    : Active
```

- Per i dettagli sull'API, vedere [DescribeSpotDatafeedSubscription](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2SpotFleetInstance

Il seguente esempio di codice mostra come utilizzare. `Get-EC2SpotFleetInstance`

Strumenti per PowerShell

Esempio 1: questo esempio descrive le istanze associate alla richiesta di flotta Spot specificata.

```
Get-EC2SpotFleetInstance -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
```

Output:

```
InstanceId   InstanceType   SpotInstanceRequestId
```

```

-----
i-f089262a    c3.large    sir-12345678
i-7e8b24a4    c3.large    sir-87654321

```

- Per i dettagli sull'API, vedere [DescribeSpotFleetInstances](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2SpotFleetRequest

Il seguente esempio di codice mostra come utilizzare. `Get-EC2SpotFleetRequest`

Strumenti per PowerShell

Esempio 1: questo esempio descrive la richiesta di flotta Spot specificata.

```

Get-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
| format-list

```

Output:

```

ConfigData      : Amazon.EC2.Model.SpotFleetRequestConfigData
CreateTime      : 12/26/2015 8:23:33 AM
SpotFleetRequestId : sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
SpotFleetRequestState : active

```

Esempio 2: questo esempio descrive tutte le richieste relative alla tua flotta Spot.

```

Get-EC2SpotFleetRequest

```

- Per i dettagli sull'API, vedere [DescribeSpotFleetRequests](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2SpotFleetRequestHistory

Il seguente esempio di codice mostra come utilizzare. `Get-EC2SpotFleetRequestHistory`

Strumenti per PowerShell

Esempio 1: questo esempio descrive la cronologia della richiesta di flotta Spot specificata.

```
Get-EC2SpotFleetRequestHistory -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE -StartTime 2015-12-26T00:00:00Z
```

Output:

```
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
```

Output:

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

- Per i dettagli sull'API, vedere [DescribeSpotFleetRequestHistory](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2SpotInstanceRequest

Il seguente esempio di codice mostra come utilizzare. Get-EC2SpotInstanceRequest

Strumenti per PowerShell

Esempio 1: questo esempio descrive la richiesta di istanza Spot specificata.

```
Get-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

Output:

```
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
```

Esempio 2: questo esempio descrive tutte le richieste di istanze Spot.

```
Get-EC2SpotInstanceRequest
```

- Per i dettagli sull'API, vedere [DescribeSpotInstanceRequests](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2SpotPriceHistory

Il seguente esempio di codice mostra come utilizzare `Get-EC2SpotPriceHistory`

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene le ultime 10 voci nella cronologia dei prezzi Spot per il tipo di istanza e la zona di disponibilità specificati. Si noti che il valore specificato per il `AvailabilityZone` parametro - deve essere valido per il valore della regione fornito al parametro `-Region` del cmdlet (non mostrato nell'esempio) o impostato come predefinito nella shell. Questo comando di esempio presuppone che nell'ambiente sia stata impostata una regione predefinita di 'us-west-2'.

```
Get-EC2SpotPriceHistory -InstanceType c3.large -AvailabilityZone us-west-2a -  
MaxResult 10
```

Output:

```
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
...
```

- Per i dettagli sull'API, vedere [DescribeSpotPriceHistory](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2Subnet

Il seguente esempio di codice mostra come utilizzare. Get-EC2Subnet

Strumenti per PowerShell

Esempio 1: questo esempio descrive la sottorete specificata.

```
Get-EC2Subnet -SubnetId subnet-1a2b3c4d
```

Output:

```
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
```

Esempio 2: questo esempio descrive tutte le sottoreti.

```
Get-EC2Subnet
```

- Per i dettagli sull'API, vedere [DescribeSubnets](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2Tag

Il seguente esempio di codice mostra come utilizzare. Get-EC2Tag

Strumenti per PowerShell

Esempio 1: questo esempio recupera i tag per il tipo di risorsa «image»

```
Get-EC2Tag -Filter @{"Name="resource-type";Values="image"}
```

Output:

Key	ResourceId	ResourceType	Value
---	-----	-----	-----
Name	ami-0a123b4ccb567a8ea	image	Win7-Imported
auto-delete	ami-0a123b4ccb567a8ea	image	never

Esempio 2: Questo esempio recupera tutti i tag per tutte le risorse e li raggruppa per tipo di risorsa

```
Get-EC2Tag | Group-Object resourcetype
```

Output:

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}

```

Esempio 3: Questo esempio mostra tutte le risorse con tag 'auto-delete' con valore 'no' per la regione data

```
Get-EC2Tag -Region eu-west-1 -Filter @{"Name"="tag:auto-delete";Values="no"}
```

Output:

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

Esempio 4: questo esempio ottiene tutte le risorse con il tag «auto-delete» con valore «no» e ulteriori filtri nella pipe successiva per analizzare solo i tipi di risorse «istanza» e infine crea il tag «» per ogni risorsa di istanza il cui valore è l'ID dell'istanza stessa ThisInstance

```

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}}

```

Esempio 5: Questo esempio recupera i tag per tutte le risorse dell'istanza e le chiavi «Name» e li visualizza in un formato tabellare

```
Get-EC2Tag -Filter @{{Name="resource-
type";Values="instance"},@{Name="key";Values="Name"}} | Select-Object ResourceId,
@{Name="Name-Tag";Expression={$PSItem.Value}} | Format-Table -AutoSize
```

Output:

```
ResourceId          Name-Tag
-----
i-012e3cb4df567e1aa jump1
i-01c23a45d6fc7a89f repro-3
```

- Per i dettagli sull'API, vedere [DescribeTags](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2Volume

Il seguente esempio di codice mostra come utilizzare `Get-EC2Volume`

Strumenti per PowerShell

Esempio 1: questo esempio descrive il volume EBS specificato.

```
Get-EC2Volume -VolumeId vol-12345678
```

Output:

```
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
```


Esempio 2: questo esempio descrive i volumi EBS con lo stato «disponibile».

```
Get-EC2Volume -Filter @{ Name="status"; Values="available" }
```

Output:

```
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
...
```

Esempio 3: questo esempio descrive tutti i tuoi volumi EBS.

```
Get-EC2Volume
```

- Per i dettagli sull'API, vedere [DescribeVolumes](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2VolumeAttribute

Il seguente esempio di codice mostra come utilizzare `Get-EC2VolumeAttribute`

Strumenti per PowerShell

Esempio 1: Questo esempio descrive l'attributo specificato del volume specificato.

```
Get-EC2VolumeAttribute -VolumeId vol-12345678 -Attribute AutoEnableIO
```

Output:

```
AutoEnableIO    ProductCodes    VolumeId
```

```
-----
False      {}      vol-12345678
```

- Per i dettagli sull'API, vedere [DescribeVolumeAttribute](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2VolumeStatus

Il seguente esempio di codice mostra come utilizzare `Get-EC2VolumeStatus`

Strumenti per PowerShell

Esempio 1: Questo esempio descrive lo stato del volume specificato.

```
Get-EC2VolumeStatus -VolumeId vol-12345678
```

Output:

```
Actions      : {}
AvailabilityZone : us-west-2a
Events       : {}
VolumeId     : vol-12345678
VolumeStatus  : Amazon.EC2.Model.VolumeStatusInfo
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus
```

Output:

```
Details                Status
-----
{io-enabled, io-performance}  ok
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus.Details
```

Output:

```
Name                Status
----
-----
```

```
io-enabled           passed
io-performance       not-applicable
```

- Per i dettagli sull'API, vedere [DescribeVolumeStatus](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2Vpc

Il seguente esempio di codice mostra come utilizzare. Get-EC2Vpc

Strumenti per PowerShell

Esempio 1: questo esempio descrive il VPC specificato.

```
Get-EC2Vpc -VpcId vpc-12345678
```

Output:

```
CidrBlock      : 10.0.0.0/16
DhcpOptionsId  : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault      : False
State          : available
Tags           : {Name}
VpcId          : vpc-12345678
```

Esempio 2: questo esempio descrive il VPC predefinito (può essercene solo uno per regione). Se il tuo account supporta EC2-Classical in questa regione, non esiste un VPC predefinito.

```
Get-EC2Vpc -Filter @{"Name"="isDefault"; Values="true"}
```

Output:

```
CidrBlock      : 172.31.0.0/16
DhcpOptionsId  : dopt-12345678
InstanceTenancy : default
IsDefault      : True
State          : available
Tags           : {}
```

```
VpcId           : vpc-45678901
```

Esempio 3: questo esempio descrive i VPC che corrispondono al filtro specificato (ovvero, hanno un CIDR che corrisponde al valore '10.0.0.0/16' e sono nello stato 'disponibile').

```
Get-EC2Vpc -Filter @{Name="cidr";  
  Values="10.0.0.0/16"},@{Name="state";Values="available"}
```

Esempio 4: questo esempio descrive tutti i tuoi VPC.

```
Get-EC2Vpc
```

- Per i dettagli sull'API, vedere [DescribeVpcs](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2VpcAttribute

Il seguente esempio di codice mostra come utilizzare. Get-EC2VpcAttribute

Strumenti per PowerShell

Esempio 1: questo esempio descrive l'attributo enableDnsSupport ".

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsSupport
```

Output:

```
EnableDnsSupport  
-----  
True
```

Esempio 2: questo esempio descrive l'attributo enableDnsHostnames ".

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsHostnames
```

Output:

```
EnableDnsHostnames  
-----
```

```
True
```

- Per i dettagli sull'API, vedere [DescribeVpcAttribute](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2VpcClassicLink

Il seguente esempio di codice mostra come utilizzare. `Get-EC2VpcClassicLink`

Strumenti per PowerShell

Esempio 1: l'esempio precedente restituisce tutti i VPC con il loro `ClassicLinkEnabled` stato per la regione

```
Get-EC2VpcClassicLink -Region eu-west-1
```

Output:

```
ClassicLinkEnabled Tags      VpcId
-----
False              {Name} vpc-0fc1ff23f45b678eb
False              {}      vpc-01e23c4a5d6db78e9
False              {Name} vpc-0123456b078b9d01f
False              {}      vpc-12cf3b4f
False              {Name} vpc-0b12d3456a7e8901d
```

- Per i dettagli sull'API, vedere [DescribeVpcClassicLink](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2VpcClassicLinkDnsSupport

Il seguente esempio di codice mostra come utilizzare. `Get-EC2VpcClassicLinkDnsSupport`

Strumenti per PowerShell

Esempio 1: Questo esempio descrive lo stato del supporto ClassicLink DNS dei VPC per la regione eu-west-1

```
Get-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

Output:

```
ClassicLinkDnsSupported VpcId
-----
False                   vpc-0b12d3456a7e8910d
False                   vpc-12cf3b4f
```

- Per i dettagli sull'API, vedere [DescribeVpcClassicLinkDnsSupportin](#) Cmdlet Reference.AWS Tools for PowerShell

Get-EC2VpcEndpoint

Il seguente esempio di codice mostra come utilizzare. `Get-EC2VpcEndpoint`

Strumenti per PowerShell

Esempio 1: questo esempio descrive uno o più endpoint VPC per la regione eu-west-1. Quindi reindirizza l'output al comando successivo, che seleziona la `VpcEndpointId` proprietà e restituisce l'ID VPC dell'array come array di stringhe

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object -ExpandProperty VpcEndpointId
```

Output:

```
vpce-01a2ab3f4f5cc6f7d
vpce-01d2b345a6787890b
vpce-0012e34d567890e12
vpce-0c123db4567890123
```

Esempio 2: Questo esempio descrive tutti gli endpoint vpc per la regione eu-west-1 e seleziona `VpcEndpointId` `ServiceName` e `PrivateDnsEnabled` proprietà per presentarla in un `VpcId` formato tabulare

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object VpcEndpointId, VpcId,
ServiceName, PrivateDnsEnabled | Format-Table -AutoSize
```

Output:

```
VpcEndpointId      VpcId      ServiceName
PrivateDnsEnabled
```

```

-----
-----
-----
vpce-02a2ab2f2f2cc2f2d vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ssm
    True
vpce-01d1b111a1114561b vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ec2
    True
vpce-0011e23d45167e838 vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ec2messages
    True
vpce-0c123db4567890123 vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ssmessages
    True

```

Esempio 3: questo esempio esporta il documento di policy per l'endpoint VPC vpce-01a2ab3f4f5cc6f7d in un file json

```
Get-EC2VpcEndpoint -Region eu-west-1 -VpcEndpointId vpce-01a2ab3f4f5cc6f7d | Select-Object -expand PolicyDocument | Out-File vpce_policyDocument.json
```

- Per i [DescribeVpcEndpoints](#) dettagli AWS Tools for PowerShell sull'API, vedere in Cmdlet Reference.

Get-EC2VpcEndpointService

Il seguente esempio di codice mostra come utilizzare. Get-EC2VpcEndpointService

Strumenti per PowerShell

Esempio 1: questo esempio descrive il servizio endpoint VPC EC2 con il filtro specificato, in questo caso com.amazonaws.eu-west-1.ecs. Inoltre, espande la proprietà e visualizza i dettagli ServiceDetails

```
Get-EC2VpcEndpointService -Region eu-west-1 -MaxResult 5 -Filter @{Name="service-name";Values="com.amazonaws.eu-west-1.ecs"} | Select-Object -ExpandProperty ServiceDetails
```

Output:

```

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

```

Esempio 2: questo esempio recupera tutti i servizi EC2 VPC Endpoint e restituisce il corrispondente «ssm» ServiceNames

```

Get-EC2VpcEndpointService -Region eu-west-1 | Select-Object -ExpandProperty
  Servicenames | Where-Object { -match "ssm"}

```

Output:

```

com.amazonaws.eu-west-1.ssm
com.amazonaws.eu-west-1.ssmmessages

```

- Per i dettagli sull'API, consulta Cmdlet Reference. [DescribeVpcEndpointServices](#) AWS Tools for PowerShell

Get-EC2VpnConnection

Il seguente esempio di codice mostra come utilizzare. Get-EC2VpnConnection

Strumenti per PowerShell

Esempio 1: questo esempio descrive la connessione VPN specificata.

```

Get-EC2VpnConnection -VpnConnectionId vpn-12345678

```

Output:

```

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
```

Esempio 2: questo esempio descrive qualsiasi connessione VPN il cui stato è in sospeso o disponibile.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnConnection -Filter $filter
```

Esempio 3: questo esempio descrive tutte le connessioni VPN.

```
Get-EC2VpnConnection
```

- Per i dettagli sull'API, vedere [DescribeVpnConnections](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EC2VpnGateway

Il seguente esempio di codice mostra come utilizzare Get-EC2VpnGateway

Strumenti per PowerShell

Esempio 1: questo esempio descrive il gateway privato virtuale specificato.

```
Get-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

Output:

```
AvailabilityZone :
State           : available
Tags            : {}
Type            : ipsec.1
VpcAttachments : {vpc-12345678}
VpnGatewayId   : vgw-1a2b3c4d
```

Esempio 2: questo esempio descrive qualsiasi gateway privato virtuale il cui stato è in sospeso o disponibile.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnGateway -Filter $filter
```

Esempio 3: questo esempio descrive tutti i gateway privati virtuali.

```
Get-EC2VpnGateway
```

- Per i dettagli sull'API, vedere [DescribeVpnGateways](#) in AWS Tools for PowerShell Cmdlet Reference.

Grant-EC2SecurityGroupEgress

Il seguente esempio di codice mostra come utilizzare `Grant-EC2SecurityGroupEgress`

Strumenti per PowerShell

Esempio 1: questo esempio definisce una regola di uscita per il gruppo di sicurezza specificato per EC2-VPC. La regola concede l'accesso all'intervallo di indirizzi IP specificato sulla porta TCP 80. La sintassi utilizzata da questo esempio richiede la PowerShell versione 3 o successiva.

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

Esempio 2: con PowerShell la versione 2, è necessario utilizzare `New-Object` per creare l'oggetto `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
```

Esempio 3: questo esempio concede l'accesso al gruppo di sicurezza di origine specificato sulla porta TCP 80.

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

Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission
@( @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; UserIdGroupPairs=$ug } )
```

- Per i dettagli sull'API, vedere [AuthorizeSecurityGroupEgress](#) in AWS Tools for PowerShell Cmdlet Reference.

Grant-EC2SecurityGroupIngress

Il seguente esempio di codice mostra come utilizzare `Grant-EC2SecurityGroupIngress`

Strumenti per PowerShell

Esempio 1: questo esempio definisce le regole di ingresso per un gruppo di sicurezza per EC2-VPC. Queste regole garantiscono l'accesso a un indirizzo IP specifico per SSH (porta 22) e RDC (porta 3389). Tieni presente che devi identificare i gruppi di sicurezza per EC2-VPC utilizzando l'ID del gruppo di sicurezza e non il nome del gruppo di sicurezza. La sintassi utilizzata da questo esempio richiede la PowerShell versione 3 o successiva.

```
$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 )
```

Esempio 2: con PowerShell la versione 2, è necessario utilizzare `New-Object` per creare gli oggetti. `IpPermission`

```
$ip1 = New-Object Amazon.EC2.Model.IpPermission
$ip1.IpProtocol = "tcp"
$ip1.FromPort = 22
$ip1.ToPort = 22
$ip1.IpRanges.Add("203.0.113.25/32")

$ip2 = new-object Amazon.EC2.Model.IpPermission
$ip2.IpProtocol = "tcp"
$ip2.FromPort = 3389
```

```
$ip2.ToPort = 3389
$ip2.IpRanges.Add("203.0.113.25/32")

Grant-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission @( $ip1, $ip2 )
```

Esempio 3: questo esempio definisce le regole di ingresso per un gruppo di sicurezza per EC2-Classic. Queste regole garantiscono l'accesso a un indirizzo IP specifico per SSH (porta 22) e RDC (porta 3389). La sintassi utilizzata da questo esempio richiede la PowerShell versione 3 o successiva.

```
$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 )
```

Esempio 4: con PowerShell la versione 2, è necessario utilizzare New-Object per creare gli oggetti. IpPermission

```
$ip1 = New-Object Amazon.EC2.Model.IpPermission
$ip1.IpProtocol = "tcp"
$ip1.FromPort = 22
$ip1.ToPort = 22
$ip1.IpRanges.Add("203.0.113.25/32")

$ip2 = new-object Amazon.EC2.Model.IpPermission
$ip2.IpProtocol = "tcp"
$ip2.FromPort = 3389
$ip2.ToPort = 3389
$ip2.IpRanges.Add("203.0.113.25/32")

Grant-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission @( $ip1,
  $ip2 )
```

Esempio 5: Questo esempio concede l'accesso alla porta TCP 8081 dal gruppo di sicurezza di origine specificato (sg-1a2b3c4d) al gruppo di sicurezza specificato (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 } )
```

Esempio 6: Questo esempio aggiunge il CIDR 5.5.5.5/32 alle regole di ingresso del gruppo di sicurezza sg-1234abcd per il traffico della porta TCP 22 con una descrizione.

```
$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
```

- Per [AuthorizeSecurityGroupIngress AWS Tools for PowerShell](#) dettagli sull'API, vedere in Cmdlet Reference.

Import-EC2Image

Il seguente esempio di codice mostra come utilizzare. Import-EC2Image

Strumenti per PowerShell

Esempio 1: questo esempio importa l'immagine di una macchina virtuale a disco singolo dal bucket Amazon S3 specificato in Amazon EC2 con un token di idempotenza. L'esempio richiede che esista un ruolo di VM Import Service con il nome predefinito «vmimport», con una policy che consenta ad Amazon EC2 l'accesso al bucket specificato, come spiegato nell'argomento VM Import Prerequisites. Per utilizzare un ruolo personalizzato, specifica il nome del ruolo utilizzando il parametro. **-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"

$params = @{
    "ClientToken"="idempotencyToken"
```

```
"Description"="Windows 2008 Standard Image Import"
"Platform"="Windows"
"LicenseType"="AWS"
}

Import-EC2Image -DiskContainer $container @parms
```

Output:

```
Architecture      :
Description       : Windows 2008 Standard Image
Hypervisor        :
ImageId           :
ImportTaskId      : import-ami-abcdefgh
LicenseType       : AWS
Platform         : Windows
Progress          : 2
SnapshotDetails   : {}
Status            : active
StatusMessage     : pending
```

- Per i dettagli sull'API, vedere [ImportImage](#) in AWS Tools for PowerShell Cmdlet Reference.

Import-EC2KeyPair

Il seguente esempio di codice mostra come utilizzare. `Import-EC2KeyPair`

Strumenti per PowerShell

Esempio 1: questo esempio importa una chiave pubblica in EC2. La prima riga memorizza il contenuto del file della chiave pubblica (*.pub) nella variabile. **\$publickey** Successivamente, l'esempio converte il formato UTF8 del file di chiave pubblica in una stringa con codifica Base64 e memorizza la stringa convertita nella variabile. **\$pkbase64** Nell'ultima riga, la chiave pubblica convertita viene importata in EC2. Il cmdlet restituisce l'impronta digitale e il nome della chiave come risultati.

```
$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
```

Output:

KeyFingerprint	KeyName
----- do:d0:15:8f:79:97:12:be:00:fd:df:31:z3:b1:42:z1	----- Example-user-key

- Per i dettagli sull'API, vedere [ImportKeyPair](#) in AWS Tools for PowerShell Cmdlet Reference.

Import-EC2Snapshot

Il seguente esempio di codice mostra come utilizzare. `Import-EC2Snapshot`

Strumenti per PowerShell

Esempio 1: questo esempio importa un'immagine del disco di una macchina virtuale in formato «VMDK» in uno snapshot di Amazon EBS. L'esempio richiede un ruolo VM Import Service con il nome predefinito 'vmimport', con una policy che consenta ad Amazon EC2 l'accesso al bucket specificato, come spiegato nell'argomento in **VM Import Prerequisites** <http://docs.aws.amazon.com/2/latest/VM.html>. **AWSEC WindowsGuide ImportPrerequisites** Per utilizzare un ruolo personalizzato, specifica il nome del ruolo utilizzando il parametro. **-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
```

Output:

Description	ImportTaskId	SnapshotTaskDetail
-----	-----	-----
Disk Image Import	import-snap-abcdefgh	Amazon.EC2.Model.SnapshotTaskDetail

- Per i dettagli sull'API, vedere [ImportSnapshot](#) in AWS Tools for PowerShell Cmdlet Reference.

Move-EC2AddressToVpc

Il seguente esempio di codice mostra come utilizzare. Move-EC2AddressToVpc

Strumenti per PowerShell

Esempio 1: questo esempio sposta un'istanza EC2 con un indirizzo IP pubblico di 12.345.67.89 sulla piattaforma EC2-VPC nella regione Stati Uniti orientali (Virginia del Nord).

```
Move-EC2AddressToVpc -PublicIp 12.345.67.89 -Region us-east-1
```

Esempio 2: questo esempio reindirizza i risultati di un comando al cmdlet. Get-EC2Instance Move-EC2AddressToVpc Il Get-EC2Instance comando ottiene un'istanza specificata dall'ID dell'istanza, quindi restituisce la proprietà dell'indirizzo IP pubblico dell'istanza.

```
(Get-EC2Instance -Instance i-12345678).Instances.PublicIpAddress | Move-EC2AddressToVpc
```

- Per i dettagli sull'API, vedere [MoveAddressToVpc](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EC2Address

Il seguente esempio di codice mostra come utilizzare. New-EC2Address

Strumenti per PowerShell

Esempio 1: questo esempio alloca un indirizzo IP elastico da utilizzare con un'istanza in un VPC.

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

Output:

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

Esempio 2: questo esempio alloca un indirizzo IP elastico da utilizzare con un'istanza in EC2-Classic.

```
New-EC2Address
```

Output:

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

- Per i dettagli sull'API, vedere [AllocateAddress](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EC2CustomerGateway

Il seguente esempio di codice mostra come utilizzare `New-EC2CustomerGateway`

Strumenti per PowerShell

Esempio 1: questo esempio crea il gateway per i clienti specificato.

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

Output:

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

- Per i dettagli sull'API, vedere [CreateCustomerGateway](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EC2DhcpOption

Il seguente esempio di codice mostra come utilizzare `New-EC2DhcpOption`

Strumenti per PowerShell

Esempio 1: Questo esempio crea il set specificato di opzioni DHCP. La sintassi utilizzata da questo esempio richiede la PowerShell versione 3 o successiva.

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

Output:

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

Esempio 2: con la PowerShell versione 2, è necessario utilizzare `New-Object` per creare ogni opzione DHCP.

```
$option1 = New-Object Amazon.EC2.Model.DhcpConfiguration
$option1.Key = "domain-name"
$option1.Values = "abc.local"

$option2 = New-Object Amazon.EC2.Model.DhcpConfiguration
$option2.Key = "domain-name-servers"
$option2.Values = @("10.0.0.101","10.0.0.102")

New-EC2DhcpOption -DhcpConfiguration @($option1, $option2)
```

Output:

```
DhcpConfigurations          DhcpOptionsId    Tags
-----
{domain-name, domain-name-servers}  dopt-2a3b4c5d    {}
```

- Per i dettagli sull'API, vedere [CreateDhcpOptions](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EC2FlowLog

Il seguente esempio di codice mostra come utilizzare `New-EC2FlowLog`

Strumenti per PowerShell

Esempio 1: questo esempio crea un flowlog EC2 per la sottorete subnet-1d234567 e il file cloud-watch-log denominato «subnet1-log» per tutto il traffico «REJECT» utilizzando le autorizzazioni del ruolo «Amministratore»

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

Output:

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

- Per i [CreateFlowLogs AWS Tools for PowerShell](#) dettagli sull'API, vedere in Cmdlet Reference.

New-EC2Host

Il seguente esempio di codice mostra come utilizzare. New-EC2Host

Strumenti per PowerShell

Esempio 1: questo esempio alloca un host dedicato all'account per il tipo di istanza e la zona di disponibilità specificati

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

Output:

```
h-01e23f4cd567890f3
```

- Per i dettagli sull'API, vedere [AllocateHosts](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EC2HostReservation

Il seguente esempio di codice mostra come utilizzare. New-EC2HostReservation

Strumenti per PowerShell

Esempio 1: questo esempio acquista l'offerta di prenotazione hro-0c1f23456789d0ab con configurazioni che corrispondono a quelle dell'host dedicato h-01e23f4cd567890f1

```
New-EC2HostReservation -OfferingId hro-0c1f23456789d0ab HostIdSet  
h-01e23f4cd567890f1
```

Output:

```
ClientToken      :  
CurrencyCode    :  
Purchase        : {hr-0123f4b5d67bedc89}  
TotalHourlyPrice : 1.307  
TotalUpfrontPrice : 0.000
```

- Per i [PurchaseHostReservation](#) dettagli AWS Tools for PowerShell sull'API, vedere in Cmdlet Reference.

New-EC2Image

Il seguente esempio di codice mostra come utilizzare. New-EC2Image

Strumenti per PowerShell

Esempio 1: questo esempio crea un AMI con il nome e la descrizione specificati, dall'istanza specificata. Amazon EC2 tenta di chiudere definitivamente l'istanza prima di creare l'immagine e riavvia l'istanza al termine.

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web  
server AMI"
```

Esempio 2: Questo esempio crea un AMI con il nome e la descrizione specificati, dall'istanza specificata. Amazon EC2 crea l'immagine senza chiudere e riavviare l'istanza; pertanto, l'integrità del file system sull'immagine creata non può essere garantita.

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web  
server AMI" -NoReboot $true
```

Esempio 3: Questo esempio crea un'AMI con tre volumi. Il primo volume è basato su uno snapshot di Amazon EBS. Il secondo volume è un volume Amazon EBS vuoto da 100 GiB. Il terzo volume è un volume di instance store. La sintassi utilizzata da questo esempio richiede la PowerShell versione 3 o successiva.

```
$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"}})
```

- Per i dettagli sull'API, vedere [CreateImage](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EC2Instance

Il seguente esempio di codice mostra come utilizzare `New-EC2Instance`

Strumenti per PowerShell

Esempio 1: questo esempio avvia una singola istanza dell'AMI specificato in EC2-Classic o un VPC predefinito.

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -InstanceType
m3.medium -KeyName my-key-pair -SecurityGroup my-security-group
```

Esempio 2: questo esempio avvia una singola istanza dell'AMI specificato in un VPC.

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -SubnetId
subnet-12345678 -InstanceType t2.micro -KeyName my-key-pair -SecurityGroupId
sg-12345678
```

Esempio 3: per aggiungere un volume EBS o un volume Instance Store, definisci una mappatura dei dispositivi a blocchi e aggiungila al comando. Questo esempio aggiunge un volume di instance store.

```
$bdm = New-Object Amazon.EC2.Model.BlockDeviceMapping
$bdm.VirtualName = "ephemeral0"
$bdm.DeviceName = "/dev/sdf"
```

```
New-EC2Instance -ImageId ami-12345678 -BlockDeviceMapping $bdm ...
```

Esempio 4: per specificare una delle AMI Windows correnti, ottieni il relativo ID AMI utilizzando `Get-EC2ImageByName`. Questo esempio avvia un'istanza dall'attuale AMI di base per Windows Server 2016.

```
$ami = Get-EC2ImageByName WINDOWS_2016_BASE  
  
New-EC2Instance -ImageId $ami.ImageId ...
```

Esempio 5: avvia un'istanza nell'ambiente host dedicato specificato.

```
New-EC2Instance -ImageId ami-1a2b3c4d -InstanceType m4.large -KeyName my-key-pair  
-SecurityGroupId sg-1a2b3c4d -AvailabilityZone us-west-1a -Tenancy host -HostID  
h-1a2b3c4d5e6f1a2b3
```

Esempio 6: questa richiesta avvia due istanze e applica alle istanze un tag con una chiave di webserver e un valore di produzione. La richiesta applica anche un tag con una chiave di cost-center e un valore cc123 ai volumi creati (in questo caso, il volume root per ogni istanza).

```
$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
```

- Per i dettagli sull'API, vedere [RunInstances](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EC2InstanceExportTask

Il seguente esempio di codice mostra come utilizzare `New-EC2InstanceExportTask`

Strumenti per PowerShell

Esempio 1: questo esempio esporta un'istanza interrotta come disco rigido virtuale (VHD) nel bucket S3. **i-0800b00a00EXAMPLE testbucket-export-instances-2019** L'ambiente di destinazione è **Microsoft**, e il parametro `region` viene aggiunto perché l'istanza si trova nella **us-east-1** regione, mentre la AWS regione predefinita dell'utente non è `us-east-1`. Per ottenere lo stato dell'attività di esportazione, copia il **ExportTaskId** valore dai risultati di questo comando, quindi esegui **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
```

Output:

```
Description           :
ExportTaskId          : export-i-077c73108aEXAMPLE
ExportToS3Task        : Amazon.EC2.Model.ExportToS3Task
InstanceExportDetails : Amazon.EC2.Model.InstanceExportDetails
State                 : active
StatusMessage         :
```

- Per i dettagli sull'API, vedere [CreateInstanceExportTask](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EC2InternetGateway

Il seguente esempio di codice mostra come utilizzare `New-EC2InternetGateway`

Strumenti per PowerShell

Esempio 1: Questo esempio crea un gateway Internet.

```
New-EC2InternetGateway
```

Output:

```
Attachments      InternetGatewayId      Tags
-----
-----
```

```
{ } igw-1a2b3c4d { }
```

- Per i dettagli sull'API, vedere [CreateInternetGateway](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EC2KeyPair

Il seguente esempio di codice mostra come utilizzare `New-EC2KeyPair`

Strumenti per PowerShell

Esempio 1: Questo esempio crea una coppia di chiavi e acquisisce la chiave privata RSA con codifica PEM in un file con il nome specificato. Quando si utilizza PowerShell, la codifica deve essere impostata su `ascii` per generare una chiave valida. Per ulteriori informazioni, consulta [Create, Display and Delete Amazon EC2 Key Pairs \(https://docs.aws.amazon.com/cli/latest/userguide/cli-services-ec2-keypairs.html\)](https://docs.aws.amazon.com/cli/latest/userguide/cli-services-ec2-keypairs.html) nella AWS Command Line Interface User Guide.

```
(New-EC2KeyPair -KeyName "my-key-pair").KeyMaterial | Out-File -Encoding ascii -FilePath C:\path\my-key-pair.pem
```

- Per i dettagli sull'API, vedere [CreateKeyPair](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EC2NetworkAcl

Il seguente esempio di codice mostra come utilizzare `New-EC2NetworkAcl`

Strumenti per PowerShell

Esempio 1: questo esempio crea un ACL di rete per il VPC specificato.

```
New-EC2NetworkAcl -VpcId vpc-12345678
```

Output:

```
Associations : {}
Entries      : {Amazon.EC2.Model.NetworkAclEntry, Amazon.EC2.Model.NetworkAclEntry}
IsDefault   : False
NetworkAclId : acl-12345678
Tags        : {}
```



```
VpcId      : vpc-12345678
```

- Per i dettagli sull'API, vedere [CreateNetworkAcl](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EC2NetworkAclEntry

Il seguente esempio di codice mostra come utilizzare `New-EC2NetworkAclEntry`

Strumenti per PowerShell

Esempio 1: Questo esempio crea una voce per l'ACL di rete specificato. La regola consente il traffico in entrata da qualsiasi luogo (0.0.0.0/0) sulla porta UDP 53 (DNS) verso qualsiasi sottorete associata.

```
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
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [CreateNetworkAclEntry](#) AWS Tools for PowerShell

New-EC2NetworkInterface

Il seguente esempio di codice mostra come utilizzare `New-EC2NetworkInterface`

Strumenti per PowerShell

Esempio 1: Questo esempio crea l'interfaccia di rete specificata.

```
New-EC2NetworkInterface -SubnetId subnet-1a2b3c4d -Description "my network  
interface" -Group sg-12345678 -PrivateIpAddress 10.0.0.17
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [CreateNetworkInterface](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EC2PlacementGroup

Il seguente esempio di codice mostra come utilizzare `New-EC2PlacementGroup`

Strumenti per PowerShell

Esempio 1: Questo esempio crea un gruppo di posizionamento con il nome specificato.

```
New-EC2PlacementGroup -GroupName my-placement-group -Strategy cluster
```

- Per i dettagli sull'API, vedere [CreatePlacementGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EC2Route

Il seguente esempio di codice mostra come utilizzare `New-EC2Route`

Strumenti per PowerShell

Esempio 1: Questo esempio crea la rotta specificata per la tabella delle rotte specificata. Il percorso corrisponde a tutto il traffico e lo invia al gateway Internet specificato.

```
New-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0 -GatewayId igw-1a2b3c4d
```

Output:

```
True
```

- Per i dettagli sull'API, vedere [CreateRoute](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EC2RouteTable

Il seguente esempio di codice mostra come utilizzare `New-EC2RouteTable`

Strumenti per PowerShell

Esempio 1: questo esempio crea una tabella di routing per il VPC specificato.

```
New-EC2RouteTable -VpcId vpc-12345678
```

Output:

```
Associations      : {}  
PropagatingVgws  : {}  
Routes           : {}  
RouteTableId     : rtb-1a2b3c4d  
Tags             : {}  
VpcId            : vpc-12345678
```

- Per i dettagli sull'API, vedere [CreateRouteTable](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EC2ScheduledInstance

Il seguente esempio di codice mostra come utilizzare `New-EC2ScheduledInstance`

Strumenti per PowerShell

Esempio 1: questo esempio avvia l'istanza pianificata specificata.

```
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
```

- Per i dettagli sull'API, vedere [RunScheduledInstances](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EC2ScheduledInstancePurchase

Il seguente esempio di codice mostra come utilizzare `New-EC2ScheduledInstancePurchase` Strumenti per PowerShell

Esempio 1: questo esempio acquista un'istanza pianificata.

```
$request = New-Object Amazon.EC2.Model.PurchaseRequest
$request.InstanceCount = 1
$request.PurchaseToken = "eyJ2IjoiMSIsInMiOiEsImMiOi..."
New-EC2ScheduledInstancePurchase -PurchaseRequest $request
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [PurchaseScheduledInstances](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EC2SecurityGroup

Il seguente esempio di codice mostra come utilizzare. New-EC2SecurityGroup

Strumenti per PowerShell

Esempio 1: questo esempio crea un gruppo di sicurezza per il VPC specificato.

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security group" -  
VpcId vpc-12345678
```

Output:

```
sg-12345678
```

Esempio 2: questo esempio crea un gruppo di sicurezza per EC2-Classic.

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security group"
```

Output:

```
sg-45678901
```

- Per i dettagli sull'API, vedere [CreateSecurityGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EC2Snapshot

Il seguente esempio di codice mostra come utilizzare. New-EC2Snapshot

Strumenti per PowerShell

Esempio 1: Questo esempio crea un'istantanea del volume specificato.

```
New-EC2Snapshot -VolumeId vol-12345678 -Description "This is a test"
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [CreateSnapshot](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EC2SpotDatafeedSubscription

Il seguente esempio di codice mostra come utilizzare `New-EC2SpotDatafeedSubscription`

Strumenti per PowerShell

Esempio 1: questo esempio crea un data feed di istanze Spot.

```
New-EC2SpotDatafeedSubscription -Bucket my-s3-bucket -Prefix spotdata
```

Output:

```
Bucket   : my-s3-bucket
Fault    :
OwnerId  : 123456789012
Prefix   : spotdata
State    : Active
```

- Per i dettagli sull'API, vedere [CreateSpotDatafeedSubscription](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EC2Subnet

Il seguente esempio di codice mostra come utilizzare `New-EC2Subnet`

Strumenti per PowerShell

Esempio 1: questo esempio crea una sottorete con il CIDR specificato.

```
New-EC2Subnet -VpcId vpc-12345678 -CidrBlock 10.0.0.0/24
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [CreateSubnet](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EC2Tag

Il seguente esempio di codice mostra come utilizzare. New-EC2Tag

Strumenti per PowerShell

Esempio 1: Questo esempio aggiunge un singolo tag alla risorsa specificata. La chiave del tag è 'myTag' e il valore del tag è 'myTagValue'. La sintassi utilizzata da questo esempio richiede la PowerShell versione 3 o successiva.

```
New-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag"; Value="myTagValue" }
```

Esempio 2: Questo esempio aggiorna o aggiunge i tag specificati alla risorsa specificata. La sintassi utilizzata da questo esempio richiede la PowerShell versione 3 o successiva.

```
New-EC2Tag -Resource i-12345678 -Tag @( @{ Key="myTag"; Value="newTagValue" },
    @{ Key="test"; Value="anotherTagValue" } )
```

Esempio 3: con PowerShell la versione 2, è necessario utilizzare New-Object per creare il tag per il parametro Tag.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

New-EC2Tag -Resource i-12345678 -Tag $tag
```

- Per i dettagli sull'API, vedere [CreateTags](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EC2Volume

Il seguente esempio di codice mostra come utilizzare `New-EC2Volume`

Strumenti per PowerShell

Esempio 1: questo esempio crea il volume specificato.

```
New-EC2Volume -Size 50 -AvailabilityZone us-west-2a -VolumeType gp2
```

Output:

```
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
```

Esempio 2: questa richiesta di esempio crea un volume e applica un tag con una chiave di pila e un valore di produzione.

```
$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
```

- Per i dettagli sull'API, vedere [CreateVolume](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EC2Vpc

Il seguente esempio di codice mostra come utilizzare. `New-EC2Vpc`

Strumenti per PowerShell

Esempio 1: questo esempio crea un VPC con il CIDR specificato. Amazon VPC crea anche quanto segue per il VPC: un set di opzioni DHCP predefinito, una tabella di routing principale e un ACL di rete predefinito.

```
New-EC2VPC -CidrBlock 10.0.0.0/16
```

Output:

```
CidrBlock      : 10.0.0.0/16
DhcpOptionsId  : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault      : False
State          : pending
Tags           : {}
VpcId          : vpc-12345678
```

- Per i dettagli sull'API, consulta [CreateVpc](#) Cmdlet Reference. AWS Tools for PowerShell

New-EC2VpcEndpoint

Il seguente esempio di codice mostra come utilizzare. `New-EC2VpcEndpoint`

Strumenti per PowerShell

Esempio 1: questo esempio crea un nuovo endpoint VPC per il servizio `com.amazonaws.eu-west-1.s3` nel VPC `vpc-0fc1ff23f45b678eb`

```
New-EC2VpcEndpoint -ServiceName com.amazonaws.eu-west-1.s3 -VpcId
vpc-0fc1ff23f45b678eb
```

Output:

```
ClientToken VpcEndpoint
-----
                Amazon.EC2.Model.VpcEndpoint
```

- Per i [CreateVpcEndpoint](#) dettagli AWS Tools for PowerShell sull'API, consulta Cmdlet Reference.

New-EC2VpnConnection

Il seguente esempio di codice mostra come utilizzare `New-EC2VpnConnection`

Strumenti per PowerShell

Esempio 1: Questo esempio crea una connessione VPN tra il gateway privato virtuale specificato e il gateway del cliente specificato. L'output include le informazioni di configurazione necessarie all'amministratore di rete, in formato XML.

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
vgw-1a2b3c4d
```

Output:

```
CustomerGatewayConfiguration : [XML document]
CustomerGatewayId           : cgw-1a2b3c4d
Options                     :
Routes                      : {}
State                       : pending
Tags                       : {}
Type                       :
VgwTelemetry               : {}
VpnConnectionId            : vpn-12345678
VpnGatewayId               : vgw-1a2b3c4d
```

Esempio 2: Questo esempio crea la connessione VPN e acquisisce la configurazione in un file con il nome specificato.

```
(New-EC2VpnConnection -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
vgw-1a2b3c4d).CustomerGatewayConfiguration | Out-File C:\path\vpn-configuration.xml
```

Esempio 3: Questo esempio crea una connessione VPN, con routing statico, tra il gateway privato virtuale specificato e il gateway del cliente specificato.

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId vgw-1a2b3c4d -Options_StaticRoutesOnly $true
```

- Per i dettagli sull'API, vedere [CreateVpnConnection](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EC2VpnConnectionRoute

Il seguente esempio di codice mostra come utilizzare. New-EC2VpnConnectionRoute

Strumenti per PowerShell

Esempio 1: questo esempio crea la route statica specificata per la connessione VPN specificata.

```
New-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock 11.12.0.0/16
```

- Per i dettagli sull'API, vedere [CreateVpnConnectionRoute](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EC2VpnGateway

Il seguente esempio di codice mostra come utilizzare. New-EC2VpnGateway

Strumenti per PowerShell

Esempio 1: questo esempio crea il gateway privato virtuale specificato.

```
New-EC2VpnGateway -Type ipsec.1
```

Output:

```
AvailabilityZone :  
State           : available  
Tags            : {}  
Type            : ipsec.1  
VpcAttachments : {}
```

```
VpnGatewayId      : vgw-1a2b3c4d
```

- Per i dettagli sull'API, vedere [CreateVpnGateway](#) in AWS Tools for PowerShell Cmdlet Reference.

Register-EC2Address

Il seguente esempio di codice mostra come utilizzare. Register-EC2Address

Strumenti per PowerShell

Esempio 1: questo esempio associa l'indirizzo IP elastico specificato all'istanza specificata in un VPC.

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

Output:

```
eipassoc-12345678
```

Esempio 2: questo esempio associa l'indirizzo IP elastico specificato all'istanza specificata in EC2-Classic.

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

- Per i dettagli sull'API, vedere [AssociateAddress](#) in AWS Tools for PowerShell Cmdlet Reference.

Register-EC2DhcpOption

Il seguente esempio di codice mostra come utilizzare. Register-EC2DhcpOption

Strumenti per PowerShell

Esempio 1: Questo esempio associa il set di opzioni DHCP specificato al VPC specificato.

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

Esempio 2: Questo esempio associa le opzioni DHCP predefinite impostate al VPC specificato.

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

- Per i dettagli sull'API, vedere [AssociateDhcpOptions](#) in AWS Tools for PowerShell Cmdlet Reference.

Register-EC2Image

Il seguente esempio di codice mostra come utilizzare. Register-EC2Image

Strumenti per PowerShell

Esempio 1: questo esempio registra un'AMI utilizzando il file manifest specificato in Amazon S3.

```
Register-EC2Image -ImageLocation my-s3-bucket/my-web-server-ami/image.manifest.xml -  
Name my-web-server-ami
```

- Per i dettagli sull'API, vedere [RegisterImage](#) in AWS Tools for PowerShell Cmdlet Reference.

Register-EC2PrivateIpAddress

Il seguente esempio di codice mostra come utilizzare. Register-EC2PrivateIpAddress

Strumenti per PowerShell

Esempio 1: Questo esempio assegna l'indirizzo IP privato secondario specificato all'interfaccia di rete specificata.

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

Esempio 2: Questo esempio crea due indirizzi IP privati secondari e li assegna all'interfaccia di rete specificata.

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

- Per i dettagli sull'API, vedere [AssignPrivateIpAddresses](#) in AWS Tools for PowerShell Cmdlet Reference.

Register-EC2RouteTable

Il seguente esempio di codice mostra come utilizzare. Register-EC2RouteTable

Strumenti per PowerShell

Esempio 1: Questo esempio associa la tabella di routing specificata alla sottorete specificata.

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

Output:

```
rtbassoc-12345678
```

- Per i dettagli sull'API, vedere [AssociateRouteTable](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-EC2Address

Il seguente esempio di codice mostra come utilizzare. Remove-EC2Address

Strumenti per PowerShell

Esempio 1: questo esempio rilascia l'indirizzo IP elastico specificato per le istanze in un VPC.

```
Remove-EC2Address -AllocationId eipalloc-12345678 -Force
```

Esempio 2: questo esempio rilascia l'indirizzo IP elastico specificato per le istanze in EC2-Classic.

```
Remove-EC2Address -PublicIp 198.51.100.2 -Force
```

- Per i dettagli sull'API, vedere [ReleaseAddress](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-EC2CapacityReservation

Il seguente esempio di codice mostra come utilizzare. Remove-EC2CapacityReservation

Strumenti per PowerShell

Esempio 1: questo esempio annulla la prenotazione di capacità cr-0c1f2345db6f7cdba

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

Output:

```
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
```

- Per i dettagli [CancelCapacityReservation AWS Tools for PowerShell](#) sull'API, vedere in Cmdlet Reference.

Remove-EC2CustomerGateway

Il seguente esempio di codice mostra come utilizzare `Remove-EC2CustomerGateway`

Strumenti per PowerShell

Esempio 1: questo esempio elimina il customer gateway specificato. Prima di procedere, viene richiesta la conferma, a meno che non si specifichi anche il parametro `Force`.

```
Remove-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

Output:

```
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"):
```

- Per i dettagli sull'API, vedere [DeleteCustomerGateway](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-EC2DhcpOption

Il seguente esempio di codice mostra come utilizzare `Remove-EC2DhcpOption`

Strumenti per PowerShell

Esempio 1: Questo esempio elimina il set di opzioni DHCP specificato. Prima di procedere con l'operazione, viene richiesta una conferma, a meno che non si specifichi anche il parametro Force.

```
Remove-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d
```

Output:

```
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"):
```

- Per i dettagli sull'API, vedere [DeleteDhcpOptions](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-EC2FlowLog

Il seguente esempio di codice mostra come utilizzare Remove-EC2FlowLog

Strumenti per PowerShell

Esempio 1: Questo esempio rimuove il dato FlowLogId fl-01a2b3456a789c01

```
Remove-EC2FlowLog -FlowLogId fl-01a2b3456a789c01
```

Output:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2FlowLog (DeleteFlowLogs)" on target
"f1-01a2b3456a789c01".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
```

- Per i dettagli AWS Tools for PowerShell sull'[DeleteFlowLogs](#) API, vedere in Cmdlet Reference.

Remove-EC2Host

Il seguente esempio di codice mostra come utilizzare. Remove-EC2Host

Strumenti per PowerShell

Esempio 1: questo esempio rilascia l'ID host specificato h-0badafd1dcb2f3456

```
Remove-EC2Host -HostId h-0badafd1dcb2f3456
```

Output:

```
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} {}
```

- Per i dettagli [ReleaseHosts AWS Tools for PowerShell](#)sull'API, vedere in Cmdlet Reference.

Remove-EC2Instance

Il seguente esempio di codice mostra come utilizzare. Remove-EC2Instance

Strumenti per PowerShell

Esempio 1: Questo esempio termina l'istanza specificata (l'istanza può essere in esecuzione o in stato «interrotto»). Il cmdlet richiederà una conferma prima di procedere; utilizzare l'opzione -Force per sopprimere la richiesta.

```
Remove-EC2Instance -InstanceId i-12345678
```

Output:

```
CurrentState                InstanceId                PreviousState
-----                -

```

```
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [TerminateInstances](#) AWS Tools for PowerShell

Remove-EC2InternetGateway

Il seguente esempio di codice mostra come utilizzare. Remove-EC2InternetGateway

Strumenti per PowerShell

Esempio 1: Questo esempio elimina il gateway Internet specificato. Prima di procedere con l'operazione, viene richiesta una conferma, a meno che non si specifichi anche il parametro Force.

```
Remove-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

Output:

```
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"):
```

- Per i dettagli sull'API, vedere [DeleteInternetGateway](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-EC2KeyPair

Il seguente esempio di codice mostra come utilizzare. Remove-EC2KeyPair

Strumenti per PowerShell

Esempio 1: Questo esempio elimina la coppia di chiavi specificata. Prima di procedere con l'operazione, viene richiesta una conferma, a meno che non si specifichi anche il parametro Force.

```
Remove-EC2KeyPair -KeyName my-key-pair
```

Output:

```
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"):
```

- Per i dettagli sull'API, vedere [DeleteKeyPair](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-EC2NetworkAcl

Il seguente esempio di codice mostra come utilizzare. Remove-EC2NetworkAcl

Strumenti per PowerShell

Esempio 1: questo esempio elimina l'ACL di rete specificato. Prima di procedere con l'operazione, viene richiesta una conferma, a meno che non si specifichi anche il parametro Force.

```
Remove-EC2NetworkAcl -NetworkAclId acl-12345678
```

Output:

```
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"):
```

- Per i dettagli sull'API, vedere [DeleteNetworkAcl](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-EC2NetworkAclEntry

Il seguente esempio di codice mostra come utilizzare. Remove-EC2NetworkAclEntry

Strumenti per PowerShell

Esempio 1: Questo esempio rimuove la regola specificata dall'ACL di rete specificato. Prima di procedere con l'operazione, viene richiesta una conferma, a meno che non si specifichi anche il parametro Force.

```
Remove-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100
```

Output:

```
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"):
```

- Per i dettagli sull'API, vedere [DeleteNetworkAclEntry](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-EC2NetworkInterface

Il seguente esempio di codice mostra come utilizzare. Remove-EC2NetworkInterface

Strumenti per PowerShell

Esempio 1: Questo esempio elimina l'interfaccia di rete specificata. Prima di procedere con l'operazione, viene richiesta una conferma, a meno che non si specifichi anche il parametro Force.

```
Remove-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

Output:

```
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"):
```

- Per i dettagli sull'API, vedere [DeleteNetworkInterface](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-EC2PlacementGroup

Il seguente esempio di codice mostra come utilizzare. Remove-EC2PlacementGroup

Strumenti per PowerShell

Esempio 1: Questo esempio elimina il gruppo di posizionamento specificato. Prima di procedere con l'operazione, viene richiesta una conferma, a meno che non si specifichi anche il parametro Force.

```
Remove-EC2PlacementGroup -GroupName my-placement-group
```

Output:

```
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"):
```

- Per i dettagli sull'API, vedere [DeletePlacementGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-EC2Route

Il seguente esempio di codice mostra come utilizzare Remove-EC2Route

Strumenti per PowerShell

Esempio 1: Questo esempio elimina la rotta specificata dalla tabella delle rotte specificata. Prima di procedere con l'operazione, viene richiesta una conferma, a meno che non si specifichi anche il parametro Force.

```
Remove-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0
```

Output:

```
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"):
```

- Per i dettagli sull'API, vedere [DeleteRoute](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-EC2RouteTable

Il seguente esempio di codice mostra come utilizzare. Remove-EC2RouteTable

Strumenti per PowerShell

Esempio 1: Questo esempio elimina la tabella di routing specificata. Prima di procedere, viene richiesta una conferma, a meno che non si specifichi anche il parametro Force.

```
Remove-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

Output:

```
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"):
```

- Per i dettagli sull'API, vedere [DeleteRouteTable](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-EC2SecurityGroup

Il seguente esempio di codice mostra come utilizzare. Remove-EC2SecurityGroup

Strumenti per PowerShell

Esempio 1: questo esempio elimina il gruppo di sicurezza specificato per EC2-VPC. Prima di procedere con l'operazione, viene richiesta una conferma, a meno che non si specifichi anche il parametro Force.

```
Remove-EC2SecurityGroup -GroupId sg-12345678
```

Output:

```
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"):
```

Esempio 2: questo esempio elimina il gruppo di sicurezza specificato per EC2-Classical.

```
Remove-EC2SecurityGroup -GroupName my-security-group -Force
```

- Per i dettagli sull'API, vedere [DeleteSecurityGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-EC2Snapshot

Il seguente esempio di codice mostra come utilizzare Remove-EC2Snapshot

Strumenti per PowerShell

Esempio 1: questo esempio elimina l'istantanea specificata. Prima di procedere con l'operazione, viene richiesta una conferma, a meno che non si specifichi anche il parametro Force.

```
Remove-EC2Snapshot -SnapshotId snap-12345678
```

Output:

```
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"):
```

- Per i dettagli sull'API, vedere [DeleteSnapshot](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-EC2SpotDatafeedSubscription

Il seguente esempio di codice mostra come utilizzare Remove-EC2SpotDatafeedSubscription

Strumenti per PowerShell

Esempio 1: questo esempio elimina il feed di dati dell'istanza Spot. Prima di procedere con l'operazione, viene richiesta una conferma, a meno che non si specifichi anche il parametro Force.

```
Remove-EC2SpotDatafeedSubscription
```

Output:

```
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"):
```

- Per i dettagli sull'API, vedere [DeleteSpotDatafeedSubscription](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-EC2Subnet

Il seguente esempio di codice mostra come utilizzare. Remove-EC2Subnet

Strumenti per PowerShell

Esempio 1: questo esempio elimina la sottorete specificata. Viene richiesta una conferma prima di procedere con l'operazione, a meno che non si specifichi anche il parametro Force.

```
Remove-EC2Subnet -SubnetId subnet-1a2b3c4d
```

Output:

```
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"):
```

- Per i dettagli sull'API, vedere [DeleteSubnet](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-EC2Tag

Il seguente esempio di codice mostra come utilizzare. Remove-EC2Tag

Strumenti per PowerShell

Esempio 1: Questo esempio elimina il tag specificato dalla risorsa specificata, indipendentemente dal valore del tag. La sintassi utilizzata da questo esempio richiede la PowerShell versione 3 o successiva.

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag" } -Force
```

Esempio 2: Questo esempio elimina il tag specificato dalla risorsa specificata, ma solo se il valore del tag corrisponde. La sintassi utilizzata da questo esempio richiede la PowerShell versione 3 o successiva.

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag";Value="myTagValue" } -Force
```

Esempio 3: Questo esempio elimina il tag specificato dalla risorsa specificata, indipendentemente dal valore del tag.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

Esempio 4: Questo esempio elimina il tag specificato dalla risorsa specificata, ma solo se il valore del tag corrisponde.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

- Per i dettagli sull'API, vedere [DeleteTags](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-EC2Volume

Il seguente esempio di codice mostra come utilizzare. Remove-EC2Volume

Strumenti per PowerShell

Esempio 1: questo esempio rimuove il volume specificato. Prima di procedere, viene richiesta una conferma, a meno che non si specifichi anche il parametro Force.

```
Remove-EC2Volume -VolumeId vol-12345678
```

Output:

```
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"):
```

- Per i dettagli sull'API, vedere [DeleteVolume](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-EC2Vpc

Il seguente esempio di codice mostra come utilizzare Remove-EC2Vpc

Strumenti per PowerShell

Esempio 1: questo esempio elimina il VPC specificato. Prima di procedere con l'operazione, viene richiesta una conferma, a meno che non si specifichi anche il parametro Force.

```
Remove-EC2Vpc -VpcId vpc-12345678
```

Output:

```
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"):
```

- Per i dettagli sull'API, vedere [DeleteVpc](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-EC2VpnConnection

Il seguente esempio di codice mostra come utilizzare Remove-EC2VpnConnection

Strumenti per PowerShell

Esempio 1: questo esempio elimina la connessione VPN specificata. Prima di procedere con l'operazione, viene richiesta una conferma, a meno che non si specifichi anche il parametro Force.

```
Remove-EC2VpnConnection -VpnConnectionId vpn-12345678
```

Output:

```
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"):
```

- Per i dettagli sull'API, vedere [DeleteVpnConnection](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-EC2VpnConnectionRoute

Il seguente esempio di codice mostra come utilizzare Remove-EC2VpnConnectionRoute

Strumenti per PowerShell

Esempio 1: questo esempio rimuove la route statica specificata dalla connessione VPN specificata. Prima di procedere con l'operazione, viene richiesta una conferma, a meno che non si specifichi anche il parametro Force.

```
Remove-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock
11.12.0.0/16
```

Output:

```
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"):
```

- Per i dettagli sull'API, vedere [DeleteVpnConnectionRoute](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-EC2VpnGateway

Il seguente esempio di codice mostra come utilizzare. Remove-EC2VpnGateway

Strumenti per PowerShell

Esempio 1: questo esempio elimina il gateway privato virtuale specificato. Prima di procedere con l'operazione, viene richiesta una conferma, a meno che non si specifichi anche il parametro Force.

```
Remove-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

Output:

```
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"):
```

- Per i dettagli sull'API, vedere [DeleteVpnGateway](#) in AWS Tools for PowerShell Cmdlet Reference.

Request-EC2SpotFleet

Il seguente esempio di codice mostra come utilizzare. Request-EC2SpotFleet

Strumenti per PowerShell

Esempio 1: questo esempio crea una richiesta di flotta Spot nella zona di disponibilità con il prezzo più basso per il tipo di istanza specificato. Se il tuo account supporta solo EC2-VPC, la flotta Spot avvia le istanze nella zona di disponibilità più economica con una sottorete predefinita. Se il tuo account supporta EC2-Classical, il parco istanze Spot lancia le istanze in EC2-Classical nella zona di disponibilità più economica. Tieni presente che il prezzo da pagare non supererà il prezzo Spot specificato per la richiesta.

```
$sg = New-Object Amazon.EC2.Model.GroupIdentifier
```

```
$sg.GroupId = "sg-12345678"
$l = New-Object Amazon.EC2.Model.SpotFleetLaunchSpecification
$l.ImageId = "ami-12345678"
$l.InstanceType = "m3.medium"
$l.SecurityGroups.Add($sg)
Request-EC2SpotFleet -SpotFleetRequestConfig_SpotPrice 0.04 `
-SpotFleetRequestConfig_TargetCapacity 2 `
-SpotFleetRequestConfig_IamFleetRole arn:aws:iam::123456789012:role/my-spot-fleet-
role `
-SpotFleetRequestConfig_LaunchSpecification $l
```

- Per i dettagli sull'API, vedere [RequestSpotFleet](#) in AWS Tools for PowerShell Cmdlet Reference.

Request-EC2SpotInstance

Il seguente esempio di codice mostra come utilizzare. Request-EC2SpotInstance

Strumenti per PowerShell

Esempio 1: questo esempio richiede un'istanza Spot una tantum nella sottorete specificata. Tieni presente che il gruppo di sicurezza deve essere creato per il VPC che contiene la sottorete specificata e deve essere specificato tramite ID utilizzando l'interfaccia di rete. Quando si specifica un'interfaccia di rete, è necessario includere l'ID di sottorete utilizzando l'interfaccia di rete.

```
$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
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [RequestSpotInstances](#) in AWS Tools for PowerShell Cmdlet Reference.

Reset-EC2ImageAttribute

Il seguente esempio di codice mostra come utilizzare. `Reset-EC2ImageAttribute`

Strumenti per PowerShell

Esempio 1: questo esempio reimposta l'attributo 'LaunchPermission' al suo valore predefinito. Per impostazione predefinita, le AMI sono private.

```
Reset-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

- Per i dettagli sull'API, vedere [ResetImageAttribute](#) in AWS Tools for PowerShell Cmdlet Reference.

Reset-EC2InstanceAttribute

Il seguente esempio di codice mostra come utilizzare. `Reset-EC2InstanceAttribute`

Strumenti per PowerShell

Esempio 1: questo esempio reimposta l'attributo 'sriovNetSupport' per l'istanza specificata.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

Esempio 2: questo esempio reimposta l'attributo 'ebsOptimized' per l'istanza specificata.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

Esempio 3: questo esempio reimposta l'attributo 'sourceDestCheck' per l'istanza specificata.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sourceDestCheck
```

Esempio 4: Questo esempio reimposta l'attributo 'disableApiTermination' per l'istanza specificata.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

Esempio 5: questo esempio reimposta l'attributo 'instanceInitiatedShutdownBehavior' per l'istanza specificata.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute  
instanceInitiatedShutdownBehavior
```

- Per i dettagli sull'API, vedere [ResetInstanceAttribute](#) in AWS Tools for PowerShell Cmdlet Reference.

Reset-EC2NetworkInterfaceAttribute

Il seguente esempio di codice mostra come utilizzare. Reset-EC2NetworkInterfaceAttribute

Strumenti per PowerShell

Esempio 1: Questo esempio reimposta il controllo di origine/destinazione per l'interfaccia di rete specificata.

```
Reset-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -SourceDestCheck
```

- Per i dettagli sull'API, vedere [ResetNetworkInterfaceAttribute](#) in Cmdlet Reference.AWS Tools for PowerShell

Reset-EC2SnapshotAttribute

Il seguente esempio di codice mostra come utilizzare. Reset-EC2SnapshotAttribute

Strumenti per PowerShell

Esempio 1: Questo esempio reimposta l'attributo specificato dell'istantanea specificata.

```
Reset-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute  
CreateVolumePermission
```

- Per i dettagli sull'API, vedere [ResetSnapshotAttribute](#) in AWS Tools for PowerShell Cmdlet Reference.

Restart-EC2Instance

Il seguente esempio di codice mostra come utilizzare `Restart-EC2Instance`

Strumenti per PowerShell

Esempio 1: questo esempio riavvia l'istanza specificata.

```
Restart-EC2Instance -InstanceId i-12345678
```

- Per i dettagli sull'API, vedere [RebootInstances](#) in AWS Tools for PowerShell Cmdlet Reference.

Revoke-EC2SecurityGroupEgress

Il seguente esempio di codice mostra come utilizzare `Revoke-EC2SecurityGroupEgress`

Strumenti per PowerShell

Esempio 1: questo esempio rimuove la regola per il gruppo di sicurezza specificato per EC2-VPC. Ciò revoca l'accesso all'intervallo di indirizzi IP specificato sulla porta TCP 80. La sintassi utilizzata da questo esempio richiede la PowerShell versione 3 o successiva.

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

Esempio 2: con PowerShell la versione 2, è necessario utilizzare `New-Object` per creare l'oggetto `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
```

Esempio 3: Questo esempio revoca l'accesso al gruppo di sicurezza di origine specificato sulla porta TCP 80.

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

- Per i dettagli sull'API, vedere [RevokeSecurityGroupEgress](#) in AWS Tools for PowerShell Cmdlet Reference.

Revoke-EC2SecurityGroupIngress

Il seguente esempio di codice mostra come utilizzare `Revoke-EC2SecurityGroupIngress`

Strumenti per PowerShell

Esempio 1: questo esempio revoca l'accesso alla porta TCP 22 dall'intervallo di indirizzi specificato per il gruppo di sicurezza specificato per EC2-VPC. Tieni presente che devi identificare i gruppi di sicurezza per EC2-VPC utilizzando l'ID del gruppo di sicurezza e non il nome del gruppo di sicurezza. La sintassi utilizzata da questo esempio richiede la PowerShell versione 3 o successiva.

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.0/24" }
Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

Esempio 2: con PowerShell la versione 2, è necessario utilizzare `New-Object` per creare l'oggetto `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
```

Esempio 3: Questo esempio revoca l'accesso alla porta TCP 22 dall'intervallo di indirizzi specificato per il gruppo di sicurezza specificato per EC2-Classic. La sintassi utilizzata da questo esempio richiede la versione 3 o successiva. PowerShell

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.0/24" }  
  
Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

Esempio 4: con PowerShell la versione 2, è necessario utilizzare New-Object per creare l'oggetto. 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
```

- Per i dettagli sull'API, vedere [RevokeSecurityGroupIngress](#) in AWS Tools for PowerShell Cmdlet Reference.

Send-EC2InstanceStatus

Il seguente esempio di codice mostra come utilizzare. Send-EC2InstanceStatus

Strumenti per PowerShell

Esempio 1: questo esempio riporta il feedback sullo stato dell'istanza specificata.

```
Send-EC2InstanceStatus -Instance i-12345678 -Status impaired -ReasonCode  
unresponsive
```

- Per i dettagli sull'API, vedere [ReportInstanceStatus](#) in AWS Tools for PowerShell Cmdlet Reference.

Set-EC2NetworkAclAssociation

Il seguente esempio di codice mostra come utilizzare. Set-EC2NetworkAclAssociation

Strumenti per PowerShell

Esempio 1: Questo esempio associa l'ACL di rete specificato alla sottorete per l'associazione ACL di rete specificata.

```
Set-EC2NetworkAclAssociation -NetworkAclId acl-12345678 -AssociationId  
aclassoc-1a2b3c4d
```

Output:

```
aclassoc-87654321
```

- Per i dettagli sull'API, vedere [ReplaceNetworkAclAssociation](#) in Cmdlet Reference.AWS Tools for PowerShell

Set-EC2NetworkAclEntry

Il seguente esempio di codice mostra come utilizzare. Set-EC2NetworkAclEntry

Strumenti per PowerShell

Esempio 1: Questo esempio sostituisce la voce specificata per l'ACL di rete specificato. La nuova regola consente il traffico in entrata dall'indirizzo specificato verso qualsiasi sottorete associata.

```
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
```

- Per i dettagli sull'API, vedere [ReplaceNetworkAclEntry](#) in AWS Tools for PowerShell Cmdlet Reference.

Set-EC2Route

Il seguente esempio di codice mostra come utilizzare. Set-EC2Route

Strumenti per PowerShell

Esempio 1: Questo esempio sostituisce il percorso specificato per la tabella di percorsi specificata. La nuova route invia il traffico specificato al gateway privato virtuale specificato.

```
Set-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 10.0.0.0/24 -GatewayId
vgw-1a2b3c4d
```

- Per i dettagli sull'API, vedere [ReplaceRoute](#) in AWS Tools for PowerShell Cmdlet Reference.

Set-EC2RouteTableAssociation

Il seguente esempio di codice mostra come utilizzare `Set-EC2RouteTableAssociation`

Strumenti per PowerShell

Esempio 1: Questo esempio associa la tabella di routing specificata alla sottorete per l'associazione della tabella di routing specificata.

```
Set-EC2RouteTableAssociation -RouteTableId rtb-1a2b3c4d -AssociationId
rtbassoc-12345678
```

Output:

```
rtbassoc-87654321
```

- Per i dettagli sull'API, vedere [ReplaceRouteTableAssociation](#) in AWS Tools for PowerShell Cmdlet Reference.

Start-EC2Instance

Il seguente esempio di codice mostra come utilizzare `Start-EC2Instance`

Strumenti per PowerShell

Esempio 1: questo esempio avvia l'istanza specificata.

```
Start-EC2Instance -InstanceId i-12345678
```

Output:

CurrentState	InstanceId	PreviousState
-----	-----	-----
Amazon.EC2.Model.InstanceState	i-12345678	Amazon.EC2.Model.InstanceState

Esempio 2: questo esempio avvia le istanze specificate.

```
@("i-12345678", "i-76543210") | Start-EC2Instance
```

Esempio 3: Questo esempio avvia il set di istanze attualmente interrotte. Gli oggetti Instance restituiti da Get-EC2Instance vengono reindirizzati a Start-EC2Instance. La sintassi utilizzata da questo esempio richiede la PowerShell versione 3 o successiva.

```
(Get-EC2Instance -Filter @{ Name="instance-state-name"; Values="stopped"}).Instances  
| Start-EC2Instance
```

Esempio 4: con la PowerShell versione 2, è necessario utilizzare New-Object per creare il filtro per il parametro Filter.

```
$filter = New-Object Amazon.EC2.Model.Filter  
$filter.Name = "instance-state-name"  
$filter.Values = "stopped"  
  
(Get-EC2Instance -Filter $filter).Instances | Start-EC2Instance
```

- Per i dettagli sull'API, vedere [StartInstances](#) in AWS Tools for PowerShell Cmdlet Reference.

Start-EC2InstanceMonitoring

Il seguente esempio di codice mostra come utilizzare Start-EC2InstanceMonitoring

Strumenti per PowerShell

Esempio 1: questo esempio consente il monitoraggio dettagliato per l'istanza specificata.

```
Start-EC2InstanceMonitoring -InstanceId i-12345678
```

Output:

```
InstanceId      Monitoring  
-----  
i-12345678     Amazon.EC2.Model.Monitoring
```

- Per i dettagli sull'API, vedere [MonitorInstances](#) in AWS Tools for PowerShell Cmdlet Reference.

Stop-EC2ImportTask

Il seguente esempio di codice mostra come utilizzare. Stop-EC2ImportTask

Strumenti per PowerShell

Esempio 1: Questo esempio annulla l'attività di importazione specificata (importazione di istantanee o immagini). Se necessario, un motivo può essere fornito utilizzando il - **CancelReason** parametro.

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

- Per i dettagli sull'API, vedere [CancelImportTask](#) in AWS Tools for PowerShell Cmdlet Reference.

Stop-EC2Instance

Il seguente esempio di codice mostra come utilizzare. Stop-EC2Instance

Strumenti per PowerShell

Esempio 1: questo esempio interrompe l'istanza specificata.

```
Stop-EC2Instance -InstanceId i-12345678
```

Output:

CurrentState	InstanceId	PreviousState
-----	-----	-----
Amazon.EC2.Model.InstanceState	i-12345678	Amazon.EC2.Model.InstanceState

- Per i dettagli sull'API, vedere [StopInstances](#) in AWS Tools for PowerShell Cmdlet Reference.

Stop-EC2InstanceMonitoring

Il seguente esempio di codice mostra come utilizzare. Stop-EC2InstanceMonitoring

Strumenti per PowerShell

Esempio 1: questo esempio disabilita il monitoraggio dettagliato per l'istanza specificata.

```
Stop-EC2InstanceMonitoring -InstanceId i-12345678
```

Output:

```
InstanceId      Monitoring
-----
i-12345678     Amazon.EC2.Model.Monitoring
```

- Per i dettagli sull'API, vedere [UnmonitorInstances](#) in AWS Tools for PowerShell Cmdlet Reference.

Stop-EC2SpotFleetRequest

Il seguente esempio di codice mostra come utilizzare. Stop-EC2SpotFleetRequest

Strumenti per PowerShell

Esempio 1: questo esempio annulla la richiesta del parco istanze Spot specificata e termina le istanze Spot associate.

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

Esempio 2: questo esempio annulla la richiesta della flotta Spot specificata senza terminare le istanze Spot associate.

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

- Per i dettagli sull'API, vedere [CancelSpotFleetRequests](#) in AWS Tools for PowerShell Cmdlet Reference.

Stop-EC2SpotInstanceRequest

Il seguente esempio di codice mostra come utilizzare. Stop-EC2SpotInstanceRequest

Strumenti per PowerShell

Esempio 1: questo esempio annulla la richiesta di istanza Spot specificata.

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

Output:

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

- Per i dettagli sull'API, vedere [CancelSpotInstanceRequests](#) in AWS Tools for PowerShell Cmdlet Reference.

Unregister-EC2Address

Il seguente esempio di codice mostra come utilizzare. Unregister-EC2Address

Strumenti per PowerShell

Esempio 1: questo esempio dissocia l'indirizzo IP elastico specificato dall'istanza specificata in un VPC.

```
Unregister-EC2Address -AssociationId eipassoc-12345678
```

Esempio 2: questo esempio dissocia l'indirizzo IP elastico specificato dall'istanza specificata in EC2-Classic.

```
Unregister-EC2Address -PublicIp 203.0.113.17
```

- Per i dettagli sull'API, vedere [DisassociateAddress](#) in Cmdlet Reference. AWS Tools for PowerShell

Unregister-EC2Image

Il seguente esempio di codice mostra come utilizzare. Unregister-EC2Image

Strumenti per PowerShell

Esempio 1: questo esempio annulla la registrazione dell'AMI specificato.

```
Unregister-EC2Image -ImageId ami-12345678
```


- Per i dettagli sull'API, vedere [DeregisterImage](#) in AWS Tools for PowerShell Cmdlet Reference.

Unregister-EC2PrivateIpAddress

Il seguente esempio di codice mostra come utilizzare `Unregister-EC2PrivateIpAddress`

Strumenti per PowerShell

Esempio 1: Questo esempio annulla l'assegnazione dell'indirizzo IP privato specificato dall'interfaccia di rete specificata.

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

- Per i dettagli sull'API, vedere [UnassignPrivateIpAddresses](#) in AWS Tools for PowerShell Cmdlet Reference.

Unregister-EC2RouteTable

Il seguente esempio di codice mostra come utilizzare `Unregister-EC2RouteTable`

Strumenti per PowerShell

Esempio 1: Questo esempio rimuove l'associazione specificata tra una tabella di routing e una sottorete.

```
Unregister-EC2RouteTable -AssociationId rtbassoc-1a2b3c4d
```

- Per i dettagli sull'API, vedere [DisassociateRouteTable](#) in AWS Tools for PowerShell Cmdlet Reference.

Esempi di Amazon ECR con Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon ECR. AWS Tools for PowerShell

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Get-ECRLoginCommand

Il seguente esempio di codice mostra come utilizzare `Get-ECRLoginCommand`.

Strumenti per PowerShell

Esempio 1: restituisce un `PSObject` contenente informazioni di accesso che possono essere utilizzate per l'autenticazione in qualsiasi registro Amazon ECR a cui il tuo principale IAM ha accesso. Le credenziali e l'endpoint della regione necessari per la chiamata per ottenere il token di autorizzazione vengono ottenuti dai valori predefiniti della shell (impostati dai cmdlet `or`). **Set-AWSCredential/Set-DefaultAWSRegion Initialize-AWSDefaultConfiguration** È possibile utilizzare la proprietà `Command` con `Invoke-Expression` per accedere al registro specificato o utilizzare le credenziali restituite in altri strumenti che richiedono l'accesso.

```
Get-ECRLoginCommand
```

Output:

```
Username      : AWS
Password      : eyJwYX1sb2Fk...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
               eyJwYX1sb2Fk...kRBVEFFS0VZIn0= https://123456789012.dkr.ecr.us-west-2.amazonaws.com
```

Esempio 2: recupera un oggetto `PSObject` contenente le informazioni di accesso che utilizzi come input per un comando `docker login`. Puoi specificare qualsiasi URI del registro Amazon ECR su cui eseguire l'autenticazione purché il tuo principale IAM abbia accesso a quel registro.

```
(Get-ECRLoginCommand).Password | docker login --username AWS --password-stdin  
012345678910.dkr.ecr.us-east-1.amazonaws.com
```

- Per i dettagli sull'API, consulta [LoginCommandGet-ECR](#) in Cmdlet Reference.AWS Tools for PowerShell

Esempi di Amazon ECS con Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon ECS. AWS Tools for PowerShell

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Get-ECSClusterDetail

Il seguente esempio di codice mostra come utilizzare `Get-ECSClusterDetail`.

Strumenti per PowerShell

Esempio 1: questo cmdlet descrive uno o più cluster ECS.

```
Get-ECSClusterDetail -Cluster "LAB-ECS-CL" -Include SETTINGS | Select-Object *
```

Output:

```
LoggedAt          : 12/27/2019 9:27:41 PM
```

```
Clusters      : {LAB-ECS-CL}
Failures      : {}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength  : 396
HttpStatusCode : OK
```

- Per i dettagli sull'API, vedere [DescribeClusters](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-ECSClusterList

Il seguente esempio di codice mostra come utilizzare. `Get-ECSClusterList`

Strumenti per PowerShell

Esempio 1: questo cmdlet restituisce un elenco di cluster ECS esistenti.

```
Get-ECSClusterList
```

Output:

```
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS-CL
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS
```

- Per i dettagli sull'API, vedere [ListClusters](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-ECSClusterService

Il seguente esempio di codice mostra come utilizzare. `Get-ECSClusterService`

Strumenti per PowerShell

Esempio 1: questo esempio elenca tutti i servizi in esecuzione nel cluster predefinito.

```
Get-ECSClusterService
```

Esempio 2: Questo esempio elenca tutti i servizi in esecuzione nel cluster specificato.

```
Get-ECSClusterService -Cluster myCluster
```

Esempio 3: Questo esempio elenca i servizi in esecuzione nel cluster specificato, recuperando un massimo di 10 dettagli del servizio alla volta.

```
$nextToken = $null
do
{
    Get-ECSClusterService -Cluster myCluster -MaxResult 10 -NextToken $nextToken
    $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Per i dettagli sull'API, vedere [ListServices](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ECSService

Il seguente esempio di codice mostra come utilizzare. `Get-ECSService`

Strumenti per PowerShell

Esempio 1: Questo esempio mostra come recuperare i dettagli di un servizio specifico dal cluster predefinito.

```
Get-ECSService -Service my-hhttp-service
```

Esempio 2: Questo esempio mostra come recuperare i dettagli di un servizio specifico in esecuzione nel cluster denominato.

```
Get-ECSService -Cluster myCluster -Service my-hhttp-service
```

- Per i dettagli sull'API, vedere [DescribeServices](#) in AWS Tools for PowerShell Cmdlet Reference.

New-ECSCluster

Il seguente esempio di codice mostra come utilizzare. `New-ECSCluster`

Strumenti per PowerShell

Esempio 1: questo cmdlet crea un nuovo cluster Amazon ECS.

```
New-ECSCluster -ClusterName "LAB-ECS-CL" -Setting @{Name="containerInsights";
Value="enabled"}
```

Output:

```

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                     : {}

```

- Per i dettagli sull'API, vedere [CreateCluster](#) in AWS Tools for PowerShell Cmdlet Reference.

New-ECSService

Il seguente esempio di codice mostra come utilizzare `New-ECSService`

Strumenti per PowerShell

Esempio 1: Questo comando di esempio crea un servizio nel cluster predefinito chiamato `ecs-simple-service`. Il servizio utilizza la definizione di attività `ecs-demo` e mantiene 10 istanze di tale attività.

```

New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10

```

Esempio 2: Questo comando di esempio crea un servizio basato su un sistema di bilanciamento del carico nel cluster predefinito chiamato `ecs-simple-service`. Il servizio utilizza la definizione di attività `ecs-demo` e mantiene 10 istanze di tale attività.

```

$lb = @{
    LoadBalancerName = "EC2Contai-EcsElast-S06278JGSJCM"
    ContainerName = "simple-demo"
    ContainerPort = 80
}

```

```
}  
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -  
DesiredCount 10 -LoadBalancer $lb
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [CreateService](#) AWS Tools for PowerShell

Remove-ECSCluster

Il seguente esempio di codice mostra come utilizzare. `Remove-ECSCluster`

Strumenti per PowerShell

Esempio 1: questo cmdlet elimina il cluster ECS specificato. È necessario annullare la registrazione di tutte le istanze del contenitore da questo cluster prima di poterlo eliminare.

```
Remove-ECSCluster -Cluster "LAB-ECS"
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DeleteCluster](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-ECSService

Il seguente esempio di codice mostra come utilizzare. `Remove-ECSService`

Strumenti per PowerShell

Esempio 1: elimina il servizio denominato 'my-http-service' nel cluster predefinito. Il servizio deve avere un conteggio desiderato e un numero corrente pari a 0 prima di poterlo eliminare. Prima di procedere con il comando, viene richiesta una conferma. Per ignorare la richiesta di conferma, aggiungere l'interruttore `-Force`.

```
Remove-ECSService -Service my-http-service
```

Esempio 2: elimina il servizio denominato 'my-http-service' nel cluster denominato.

```
Remove-ECSService -Cluster myCluster -Service my-http-service
```

- Per i dettagli sull'API, vedere [DeleteService](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-ECSClusterSetting

Il seguente esempio di codice mostra come utilizzare. Update-ECSClusterSetting

Strumenti per PowerShell

Esempio 1: questo cmdlet modifica le impostazioni da utilizzare per un cluster ECS.

```
Update-ECSClusterSetting -Cluster "LAB-ECS-CL" -Setting @{Name="containerInsights";  
Value="disabled"}
```

Output:

```
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                     : {}
```

- Per i dettagli sull'API, vedere [UpdateClusterSettings](#) in Cmdlet Reference.AWS Tools for PowerShell

Update-ECSService

Il seguente esempio di codice mostra come utilizzare. Update-ECSService

Strumenti per PowerShell

Esempio 1: questo comando di esempio aggiorna il servizio my-http-service `` per utilizzare la definizione del task amazon-ecs-sample ``.

```
Update-ECSService -Service my-http-service -TaskDefinition amazon-ecs-sample
```

Esempio 2: questo comando di esempio aggiorna il conteggio desiderato del servizio my-http-service `` a 10.

```
Update-ECSService -Service my-http-service -DesiredCount 10
```

- Per i dettagli sull'API, vedere [UpdateService](#) in AWS Tools for PowerShell Cmdlet Reference.

Esempi di Amazon EFS con Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon EFS. AWS Tools for PowerShell

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Edit-EFSMountTargetSecurityGroup

Il seguente esempio di codice mostra come utilizzare `Edit-EFSMountTargetSecurityGroup`.

Strumenti per PowerShell

Esempio 1: aggiorna i gruppi di sicurezza in vigore per il target di montaggio specificato. È possibile specificarne fino a 5, nel formato «sg-xxxxxxx».

```
Edit-EFSMountTargetSecurityGroup -MountTargetId fsmt-1a2b3c4d -SecurityGroup sg-group1,sg-group3
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [ModifyMountTargetSecurityGroups](#) AWS Tools for PowerShell

Get-EFSFileSystem

Il seguente esempio di codice mostra come utilizzare. Get-EFSFileSystem

Strumenti per PowerShell

Esempio 1: restituisce la raccolta di tutti i file system di proprietà dell'account del chiamante nella regione.

```
Get-EFSFileSystem
```

Output:

```
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
...
```

Esempio 2: restituisce i dettagli del file system specificato.

```
Get-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

Esempio 3: restituisce i dettagli di un file system utilizzando il token di creazione dell'idempotenza specificato al momento della creazione del file system.

```
Get-EFSFileSystem -CreationToken 1a2bff54-85e0-4747-bd95-7bc172c4f555
```

- Per i dettagli sull'API, vedere [DescribeFileSystems](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EFSMountTarget

Il seguente esempio di codice mostra come utilizzare. `Get-EFSMountTarget`

Strumenti per PowerShell

Esempio 1: restituisce la raccolta di destinazioni di montaggio associate al file system specificato.

```
Get-EFSMountTarget -FileSystemId fs-1a2b3c4d
```

Output:

```
FileSystemId      : fs-1a2b3c4d
IpAddress         : 10.0.0.131
LifeCycleState   : available
MountTargetId    : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId          : 123456789012
SubnetId         : subnet-1a2b3c4d
```

- Per i dettagli sull'API, vedere [DescribeMountTargets](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EFSMountTargetSecurityGroup

Il seguente esempio di codice mostra come utilizzare. `Get-EFSMountTargetSecurityGroup`

Strumenti per PowerShell

Esempio 1: restituisce gli ID dei gruppi di sicurezza attualmente assegnati all'interfaccia di rete associata al target di montaggio.

```
Get-EFSMountTargetSecurityGroup -MountTargetId fsmt-1a2b3c4d
```

Output:

```
sg-1a2b3c4d
```

- Per i dettagli sull'API, vedere [DescribeMountTargetSecurityGroups](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EFSTag

Il seguente esempio di codice mostra come utilizzare. Get-EFSTag

Strumenti per PowerShell

Esempio 1: restituisce la raccolta di tag attualmente associati al file system specificato.

```
Get-EFSTag -FileSystemId fs-1a2b3c4d
```

Output:

Key	Value
---	-----
Name	My File System
tagkey1	tagvalue1
tagkey2	tagvalue2

- Per i dettagli sull'API, vedere [DescribeTags](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EFSFileSystem

Il seguente esempio di codice mostra come utilizzare. New-EFSFileSystem

Strumenti per PowerShell

Esempio 1: crea un nuovo file system vuoto. Il token utilizzato per garantire una creazione idempotente verrà generato automaticamente ed è accessibile dal **CreationToken** membro dell'oggetto restituito.

```
New-EFSFileSystem
```

Output:

```
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
```

Esempio 2: crea un nuovo file system vuoto utilizzando un token personalizzato per garantire una creazione idempotente.

```
New-EFSFileSystem -CreationToken "MyUniqueToken"
```

- Per i dettagli sull'API, vedere [CreateFileSystem](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EFSMountTarget

Il seguente esempio di codice mostra come utilizzare `New-EFSMountTarget`

Strumenti per PowerShell

Esempio 1: crea una nuova destinazione di montaggio per un file system. La sottorete specificata verrà utilizzata per determinare il Virtual Private Cloud (VPC) in cui verrà creato il mount target e l'indirizzo IP che verrà assegnato automaticamente (dall'intervallo di indirizzi della sottorete). L'indirizzo IP assegnato può essere utilizzato per poi montare questo file system su un'istanza Amazon EC2. Poiché non è stato specificato alcun gruppo di sicurezza, l'interfaccia di rete creata per la destinazione è associata al gruppo di sicurezza predefinito per il VPC della sottorete.

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d
```

Output:

```
FileSystemId      : fs-1a2b3c4d
IpAddress         : 10.0.0.131
LifecycleState    : creating
```

```
MountTargetId      : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId            : 123456789012
SubnetId           : subnet-1a2b3c4d
```

Esempio 2: crea una nuova destinazione di montaggio per il file system specificato con indirizzo IP assegnato automaticamente. L'interfaccia di rete creata per la destinazione di montaggio è associata ai gruppi di sicurezza specificati (è possibile specificare fino a 5, nel formato «sg-xxxxxxx»).

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -
SecurityGroup sg-group1,sg-group2,sg-group3
```

Esempio 3: crea una nuova destinazione di montaggio per il file system specificato con l'indirizzo IP specificato.

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -IpAddress
10.0.0.131
```

- Per i dettagli sull'API, vedere [CreateMountTarget](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EFSTag

Il seguente esempio di codice mostra come utilizzare New-EFSTag

Strumenti per PowerShell

Esempio 1: applica la raccolta di tag al file system specificato. Se un tag con la chiave specificata esiste già nel file system, il valore del tag viene aggiornato.

```
New-EFSTag -FileSystemId fs-1a2b3c4d -Tag
@{Key="tagkey1";Value="tagvalue1"},@{Key="tagkey2";Value="tagvalue2"}
```

Esempio 2: imposta il name tag per il file system specificato. Questo valore viene restituito insieme ad altri dettagli del file system quando viene utilizzato il FileSystem cmdlet Get-EFS.

```
New-EFSTag -FileSystemId fs-1a2b3c4d -Tag @{Key="Name";Value="My File System"}
```

- Per i dettagli sull'API, vedere [CreateTags](#) in Cmdlet Reference.AWS Tools for PowerShell

Remove-EFSFileSystem

Il seguente esempio di codice mostra come utilizzare. Remove-EFSFileSystem

Strumenti per PowerShell

Esempio 1: elimina il file system specificato che non è più in uso (se il file system ha destinazioni di montaggio, queste devono essere prima rimosse). Prima di procedere con il cmdlet, viene richiesta la conferma. Per eliminare la conferma, utilizzare lo switch. **-Force**

```
Remove-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [DeleteFileSystem](#) AWS Tools for PowerShell

Remove-EFSMountTarget

Il seguente esempio di codice mostra come utilizzare. Remove-EFSMountTarget

Strumenti per PowerShell

Esempio 1: elimina il target di montaggio specificato. Prima di procedere con l'operazione, viene richiesta una conferma. Per sopprimere la richiesta, utilizzare l'interruttore. **-Force** Notate che questa operazione interrompe in modo forzato qualsiasi installazione del file system tramite la destinazione: se possibile, potreste prendere in considerazione la possibilità di smontare il file system prima di eseguire questo comando.

```
Remove-EFSMountTarget -MountTargetId fsmt-1a2b3c4d
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [DeleteMountTarget](#) AWS Tools for PowerShell

Remove-EFSTag

Il seguente esempio di codice mostra come utilizzare. Remove-EFSTag

Strumenti per PowerShell

Esempio 1: elimina la raccolta di uno o più tag da un file system. Prima di procedere con il cmdlet, viene richiesta la conferma. Per eliminare la conferma, utilizzare lo switch. **-Force**

```
Remove-EFSTag -FileSystemId fs-1a2b3c4d -TagKey "tagkey1","tagkey2"
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [DeleteTags](#) AWS Tools for PowerShell

Esempi di Amazon EKS con Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell con Amazon EKS.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Add-EKSResourceTag

Il seguente esempio di codice mostra come utilizzare `Add-EKSResourceTag`.

Strumenti per PowerShell

Esempio 1: questo cmdlet associa i tag specificati a una risorsa con il ResourceArn specificato.

```
Add-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD" -  
Tag @{Name = "EKSPRODCLUSTER"}
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [TagResource](#) AWS Tools for PowerShell

Get-EKSCluster

Il seguente esempio di codice mostra come utilizzare. Get-EKSCluster

Strumenti per PowerShell

Esempio 1: questo cmdlet restituisce informazioni descrittive su un cluster Amazon EKS.

```
Get-EKSCluster -Name "PROD"
```

Output:

```
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
```

- Per i dettagli sull'API, consulta Cmdlet [DescribeCluster](#) Reference AWS Tools for PowerShell .

Get-EKSClusterList

Il seguente esempio di codice mostra come utilizzare. Get-EKSClusterList

Strumenti per PowerShell

Esempio 1: questo cmdlet elenca i cluster Amazon EKS presenti Account AWS nella regione specificata.

```
Get-EKSClusterList
```

Output:

```
PROD
```

- Per i dettagli sull'API, vedere [ListClusters](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EKSFargateProfile

Il seguente esempio di codice mostra come utilizzare `Get-EKSFargateProfile`

Strumenti per PowerShell

Esempio 1: questo cmdlet restituisce informazioni descrittive su un profilo AWS Fargate.

```
Get-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

Output:

```
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             : {}
```

- Per i dettagli sull'API, vedere [DescribeFargateProfile](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-EKSFargateProfileList

Il seguente esempio di codice mostra come utilizzare `Get-EKSFargateProfileList`

Strumenti per PowerShell

Esempio 1: questo cmdlet elenca i profili AWS Fargate associati al cluster specificato Account AWS nella regione specificata.

```
Get-EKSFargateProfileList -ClusterName "TEST"
```

Output:

```
EKSFargate  
EKSFargateProfile
```

- Per i dettagli sull'API, vedere [ListFargateProfiles](#) in AWS Tools for PowerShell Cmdlet Reference.

Get -EKSNodegroup

Il seguente esempio di codice mostra come utilizzare. Get -EKSNodegroup

Strumenti per PowerShell

Esempio 1: questo cmdlet restituisce informazioni descrittive su un gruppo di nodi Amazon EKS.

```
Get-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

Output:

```
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
```

- Per i dettagli sull'API, consulta Cmdlet [DescribeNodegroup](#) Reference AWS Tools for PowerShell .

Get-EKSNodegroupList

Il seguente esempio di codice mostra come utilizzare. `Get-EKSNodegroupList`

Strumenti per PowerShell

Esempio 1: questo cmdlet elenca i gruppi di nodi Amazon EKS associati al cluster specificato Account AWS nella regione specificata.

```
Get-EKSNodegroupList -ClusterName PROD
```

Output:

```
ProdEKSNodeGroup
```

- Per i dettagli sull'API, vedere [ListNodegroups](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-EKSResourceTag

Il seguente esempio di codice mostra come utilizzare. `Get-EKSResourceTag`

Strumenti per PowerShell

Esempio 1: questo cmdlet elenca i tag per una risorsa Amazon EKS.

```
Get-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
```

Output:

```
Key Value
---
Name EKSPRODCLUSTER
```

- Per i dettagli sull'API, consulta AWS Tools for PowerShell Cmdlet [ListTagsForResource](#) Reference.

Get-EKSUpdate

Il seguente esempio di codice mostra come utilizzare. `Get-EKSUpdate`

Strumenti per PowerShell

Esempio 1: questo cmdlet restituisce informazioni descrittive su un aggiornamento del cluster Amazon EKS o del gruppo di nodi gestiti associato.

```
Get-EKSUpdate -Name "PROD" -UpdateId "ee708232-7d2e-4ed7-9270-d0b5176f0726"
```

Output:

```
CreatedAt : 12/25/2019 5:03:07 PM
Errors    : {}
Id        : ee708232-7d2e-4ed7-9270-d0b5176f0726
Params    : {Amazon.EKS.Model.UpdateParam}
Status    : Successful
Type      : LoggingUpdate
```

- Per i dettagli sull'API, consulta Cmdlet [DescribeUpdate](#) Reference AWS Tools for PowerShell .

Get-EKSUpdateList

Il seguente esempio di codice mostra come utilizzare. `Get-EKSUpdateList`

Strumenti per PowerShell

Esempio 1: questo cmdlet elenca gli aggiornamenti associati a un cluster Amazon EKS o a un gruppo di nodi gestiti nella tua regione Account AWS, nella regione specificata.

```
Get-EKSUpdateList -Name "PROD"
```

Output:

```
ee708232-7d2e-4ed7-9270-d0b5176f0726
```

- Per i dettagli sull'API, consulta AWS Tools for PowerShell Cmdlet [ListUpdates](#) Reference.

New-EKSCluster

Il seguente esempio di codice mostra come utilizzare `New-EKSCluster`

Strumenti per PowerShell

Esempio 1: Questo esempio crea un nuovo cluster chiamato 'prod'.

```
New-EKSCluster -Name prod -ResourcesVpcConfig
@{SubnetIds=@("subnet-0a1b2c3d","subnet-3a2b1c0d");SecurityGroupIds="sg-6979fe18"}
-RoleArn "arn:aws:iam::012345678901:role/eks-service-role"
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [CreateCluster](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EKSFargateProfile

Il seguente esempio di codice mostra come utilizzare `New-EKSFargateProfile`

Strumenti per PowerShell

Esempio 1: questo cmdlet crea un profilo AWS Fargate per il tuo cluster Amazon EKS. È necessario disporre di almeno un profilo Fargate in un cluster per poter pianificare i pod sull'infrastruttura Fargate.

```
New-EKSFargateProfile -FargateProfileName EKSFargateProfile -ClusterName TEST -
Subnet "subnet-02f6ff500ff2067a0", "subnet-0cd976f08d5fbfaae" -PodExecutionRoleArn
arn:aws:iam::012345678912:role/AmazonEKSFargatePodExecutionRole -Selector
@{Namespace="default"}
```

Output:

```

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             : {}

```

- Per i dettagli sull'API, vedere [CreateFargateProfile](#) in AWS Tools for PowerShell Cmdlet Reference.

New-EKSNodeGroup

Il seguente esempio di codice mostra come utilizzare. New-EKSNodeGroup

Strumenti per PowerShell

Esempio 1: questo cmdlet crea un gruppo di nodi di lavoro gestito per un cluster Amazon EKS. È possibile creare solo un gruppo di nodi per il cluster che sia uguale alla versione corrente di Kubernetes per il cluster. Tutti i gruppi di nodi vengono creati con l'ultima versione AMI per la rispettiva versione Kubernetes secondaria del cluster.

```

New-EKSNodeGroup -NodeGroupName "ProdEKSNodeGroup" -AmiType "AL2_x86_64"
-DiskSize 40 -ClusterName "PROD" -ScalingConfig_DesiredSize 2 -
ScalingConfig_MinSize 2 -ScalingConfig_MaxSize 5 -InstanceType t3.large
-NodeRole "arn:aws:iam::012345678912:role/NodeInstanceRole" -Subnet
"subnet-0d1a9fff35efa7691","subnet-0a3f4928edbc224d4"

```

Output:

```

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

```

- Per i dettagli sull'API, vedere [CreateNodegroup](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-EKSCluster

Il seguente esempio di codice mostra come utilizzare `Remove-EKSCluster`

Strumenti per PowerShell

Esempio 1: questo cmdlet elimina il piano di controllo del cluster Amazon EKS.

```
Remove-EKSCluster -Name "DEV-KUBE-CL"
```

Output:

```

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.y14.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

```

- Per i dettagli sull'API, consulta Cmdlet [DeleteCluster](#) Reference AWS Tools for PowerShell .

Remove-EKSFargateProfile

Il seguente esempio di codice mostra come utilizzare. Remove-EKSFargateProfile

Strumenti per PowerShell

Esempio 1: questo cmdlet elimina un profilo AWS Fargate. Quando si elimina un profilo Fargate, tutti i pod in esecuzione su Fargate creati con il profilo vengono eliminati.

```
Remove-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

Output:

```

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 : {}
```

- Per i dettagli sull'API, vedere [DeleteFargateProfile](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-EKSNodegroup

Il seguente esempio di codice mostra come utilizzare `Remove-EKSNodegroup`

Strumenti per PowerShell

Esempio 1: questo cmdlet elimina un gruppo di nodi Amazon EKS per un cluster.

```
Remove-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

Output:

```
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
```

- Per i dettagli sull'API, consulta Cmdlet [DeleteNodegroup](#) Reference AWS Tools for PowerShell .

Remove-EKSResourceTag

Il seguente esempio di codice mostra come utilizzare. Remove-EKSResourceTag

Strumenti per PowerShell

Esempio 1: questo cmdlet elimina i tag specificati da una risorsa EKS.

```
Remove-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
-TagKey "Name"
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [UntagResource](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-EKSClusterConfig

Il seguente esempio di codice mostra come utilizzare. Update-EKSClusterConfig

Strumenti per PowerShell

Esempio 1: aggiorna una configurazione del cluster Amazon EKS. Il cluster continua a funzionare durante l'aggiornamento.

```
Update-EKSClusterConfig -Name "PROD" -Logging_ClusterLogging
@{Types="api","audit","authenticator","controllerManager","scheduler",Enabled="True"}
```

Output:

```
CreatedAt : 12/25/2019 5:03:07 PM
Errors    : {}
Id        : ee708232-7d2e-4ed7-9270-d0b5176f0726
Params    : {Amazon.EKS.Model.UpdateParam}
Status    : InProgress
Type      : LoggingUpdate
```

- Per i dettagli sull'API, vedere [UpdateClusterConfig](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-EKSClusterVersion

Il seguente esempio di codice mostra come utilizzare `Update-EKSClusterVersion`

Strumenti per PowerShell

Esempio 1: questo cmdlet aggiorna un cluster Amazon EKS alla versione Kubernetes specificata. Il cluster continua a funzionare durante l'aggiornamento.

```
Update-EKSClusterVersion -Name "PROD-KUBE-CL" -Version 1.14
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [UpdateClusterVersion](#) in AWS Tools for PowerShell Cmdlet Reference.

Elastic Load Balancing - Esempi della versione 1 che utilizzano Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with Elastic Load Balancing - Versione 1.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Add-ELBLoadBalancerToSubnet

Il seguente esempio di codice mostra come utilizzare `Add-ELBLoadBalancerToSubnet`.

Strumenti per PowerShell

Esempio 1: Questo esempio aggiunge la sottorete specificata all'insieme di sottoreti configurate per il sistema di bilanciamento del carico specificato. L'output include l'elenco completo delle sottoreti.

```
Add-ELBLoadBalancerToSubnet -LoadBalancerName my-load-balancer -Subnet
subnet-12345678
```

Output:

```
subnet-12345678
subnet-87654321
```

- Per i dettagli sull'API, vedere [AttachLoadBalancerToSubnets](#) in AWS Tools for PowerShell Cmdlet Reference.

Add-ELBResourceTag

Il seguente esempio di codice mostra come utilizzare `Add-ELBResourceTag`

Strumenti per PowerShell

Esempio 1: Questo esempio aggiunge i tag specificati al sistema di bilanciamento del carico specificato. La sintassi utilizzata da questo esempio richiede la PowerShell versione 3 o successiva.

```
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag
@{ Key="project";Value="lima" },@{ Key="department";Value="digital-media" }
```

Esempio 2: con PowerShell la versione 2, è necessario utilizzare New-Object per creare un tag per il parametro Tag.

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.Tag
$tag.Key = "project"
$tag.Value = "lima"
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag $tag
```

- Per i dettagli sull'API, vedere [AddTags](#) in AWS Tools for PowerShell Cmdlet Reference.

Disable-ELBAvailabilityZoneForLoadBalancer

Il seguente esempio di codice mostra come utilizzare. Disable-ELBAvailabilityZoneForLoadBalancer

Strumenti per PowerShell

Esempio 1: Questo esempio rimuove la zona di disponibilità specificata dal sistema di bilanciamento del carico specificato. L'output include le zone di disponibilità rimanenti.

```
Disable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -
AvailabilityZone us-west-2a
```

Output:

```
us-west-2b
```

- Per i dettagli sull'API, vedere [DisableAvailabilityZonesForLoadBalancer](#) in AWS Tools for PowerShell Cmdlet Reference.

Dismount-ELBLoadBalancerFromSubnet

Il seguente esempio di codice mostra come utilizzare `Dismount-ELBLoadBalancerFromSubnet`

Strumenti per PowerShell

Esempio 1: Questo esempio rimuove la sottorete specificata dal set di sottoreti configurate per il sistema di bilanciamento del carico specificato. L'output include le sottoreti rimanenti.

```
Dismount-ELBLoadBalancerFromSubnet -LoadBalancerName my-load-balancer -Subnet subnet-12345678
```

Output:

```
subnet-87654321
```

- Per i dettagli sull'API, vedere [DetachLoadBalancerFromSubnets](#) in AWS Tools for PowerShell Cmdlet Reference.

Edit-ELBLoadBalancerAttribute

Il seguente esempio di codice mostra come utilizzare `Edit-ELBLoadBalancerAttribute`

Strumenti per PowerShell

Esempio 1: questo esempio abilita il bilanciamento del carico tra zone per il bilanciamento del carico specificato.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -CrossZoneLoadBalancing_Enabled $true
```

Esempio 2: questo esempio disabilita il drenaggio delle connessioni per il sistema di bilanciamento del carico specificato.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -ConnectionDraining_Enabled $false
```

Esempio 3: questo esempio abilita la registrazione degli accessi per il sistema di bilanciamento del carico specificato.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer `
```

```
>> -AccessLog_Enabled $true `
>> -AccessLog_S3BucketName my-logs-bucket `
>> -AccessLog_S3BucketPrefix my-app/prod `
>> -AccessLog_EmitInterval 60
```

- Per i dettagli sull'API, vedere [ModifyLoadBalancerAttributes](#) in AWS Tools for PowerShell Cmdlet Reference.

Enable-ELBAvailabilityZoneForLoadBalancer

Il seguente esempio di codice mostra come utilizzare. `Enable-ELBAvailabilityZoneForLoadBalancer`

Strumenti per PowerShell

Esempio 1: Questo esempio aggiunge la zona di disponibilità specificata al sistema di bilanciamento del carico specificato. L'output include l'elenco completo delle zone di disponibilità.

```
Enable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -
AvailabilityZone us-west-2a
```

Output:

```
us-west-2a
us-west-2b
```

- Per i dettagli sull'API, vedere [EnableAvailabilityZonesForLoadBalancer](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ELBInstanceHealth

Il seguente esempio di codice mostra come utilizzare. `Get-ELBInstanceHealth`

Strumenti per PowerShell

Esempio 1: Questo esempio descrive lo stato delle istanze registrate con il sistema di bilanciamento del carico specificato.

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer
```


Output:

Description	InstanceId	ReasonCode
State		
-----	-----	-----

N/A	i-87654321	N/A
InService		
Instance has failed at lea...	i-12345678	Instance
OutOfService		

Esempio 2: questo esempio descrive lo stato dell'istanza specificata registrata con il sistema di bilanciamento del carico specificato.

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance i-12345678
```

Esempio 3: questo esempio visualizza la descrizione completa dello stato dell'istanza specificata.

```
(Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance i-12345678).Description
```

Output:

```
Instance has failed at least the UnhealthyThreshold number of health checks consecutively.
```

- Per i dettagli sull'API, vedere [DescribeInstanceHealth](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ELBLoadBalancer

Il seguente esempio di codice mostra come utilizzare `Get-ELBLoadBalancer`

Strumenti per PowerShell

Esempio 1: questo esempio elenca i nomi dei sistemi di bilanciamento del carico.

```
Get-ELBLoadBalancer | format-table -property LoadBalancerName
```

Output:

```
LoadBalancerName
-----
my-load-balancer
my-other-load-balancer
my-internal-load-balancer
```

Esempio 2: questo esempio descrive il sistema di bilanciamento del carico specificato.

```
Get-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

Output:

```
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
```

Esempio 3: questo esempio descrive tutti i sistemi di bilanciamento del carico nella regione corrente. AWS

```
Get-ELBLoadBalancer
```

Esempio 4: Questo esempio descrive tutti i sistemi di bilanciamento del carico tra tutti quelli disponibili. Regioni AWS

```
Get-AWSRegion | % { Get-ELBLoadBalancer -Region $_ }
```

- Per i dettagli sull'API, vedere [DescribeLoadBalancers](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ELBLoadBalancerAttribute

Il seguente esempio di codice mostra come utilizzare. `Get-ELBLoadBalancerAttribute`

Strumenti per PowerShell

Esempio 1: questo esempio descrive gli attributi del load balancer specificato.

```
Get-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer
```

Output:

```
AccessLog           : Amazon.ElasticLoadBalancing.Model.AccessLog
AdditionalAttributes : {}
ConnectionDraining  : Amazon.ElasticLoadBalancing.Model.ConnectionDraining
ConnectionSettings  : Amazon.ElasticLoadBalancing.Model.ConnectionSettings
CrossZoneLoadBalancing : Amazon.ElasticLoadBalancing.Model.CrossZoneLoadBalancing
```

- Per i dettagli sull'API, vedere [DescribeLoadBalancerAttributes](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ELBLoadBalancerPolicy

Il seguente esempio di codice mostra come utilizzare. `Get-ELBLoadBalancerPolicy`

Strumenti per PowerShell

Esempio 1: Questo esempio descrive le politiche associate al sistema di bilanciamento del carico specificato.

```
Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer
```

Output:

```
PolicyAttributeDescriptions      PolicyName
PolicyTypeName
```

```

-----
-----
{ProxyProtocol}                my-ProxyProtocol-policy
ProxyProtocolPolicyType
{CookieName}                   my-app-cookie-policy
AppCookieStickinessPolicyType

```

Esempio 2: questo esempio descrive gli attributi della politica specificata.

```
(Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-ProxyProtocol-policy).PolicyAttributeDescriptions
```

Output:

```

AttributeName      AttributeValue
-----
ProxyProtocol      true

```

Esempio 3: Questo esempio descrive le politiche predefinite, incluse le politiche di esempio. I nomi delle politiche di esempio hanno il prefisso elbSample-.

```
Get-ELBLoadBalancerPolicy
```

Output:

```

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
```

- Per i dettagli sull'API, vedere [DescribeLoadBalancerPolicies](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ELBLoadBalancerPolicyType

Il seguente esempio di codice mostra come utilizzare `Get-ELBLoadBalancerPolicyType`

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene i tipi di policy supportati da Elastic Load Balancing.

```
Get-ELBLoadBalancerPolicyType
```

Output:

Description	PolicyAttributeTypeDescriptions
PolicyTypeName	-----
-----	-----
Stickiness policy with session lifet... LBCookieStickinessPolicyType	{CookieExpirationPeriod}
Policy that controls authentication ... BackendServerAuthenticationPolicyType	{PublicKeyPolicyName}
Listener policy that defines the cip... SSLNegotiationPolicyType	{Protocol-SSLv2, Protocol-TLSv1, Pro...
Policy containing a list of public k... PublicKeyPolicyType	{PublicKey}
Stickiness policy with session lifet... AppCookieStickinessPolicyType	{CookieName}
Policy that controls whether to incl... ProxyProtocolPolicyType	{ProxyProtocol}

Esempio 2: Questo esempio descrive il tipo di policy specificato.

```
Get-ELBLoadBalancerPolicyType -PolicyTypeName ProxyProtocolPolicyType
```

Output:

```

Description                                PolicyAttributeTypeDescriptions
PolicyTypeName
-----
-----
Policy that controls whether to incl... {ProxyProtocol}
ProxyProtocolPolicyType

```

Esempio 3: Questo esempio visualizza la descrizione completa del tipo di politica specificato.

```
(Get-ELBLoadBalancerPolicyType -PolicyTypeName).Description
```

Output:

```

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

```

- Per i dettagli sull'API, vedere [DescribeLoadBalancerPolicyTypes](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ELBResourceTag

Il seguente esempio di codice mostra come utilizzare. Get-ELBResourceTag

Strumenti per PowerShell

Esempio 1: questo esempio elenca i tag per i sistemi di bilanciamento del carico specificati.

```
Get-ELBResourceTag -LoadBalancerName @("my-load-balancer","my-internal-load-
balancer")
```

Output:

```

LoadBalancerName      Tags
-----
my-load-balancer      {project, department}
my-internal-load-balancer {project, department}

```

Esempio 2: Questo esempio descrive i tag per il sistema di bilanciamento del carico specificato.

```
(Get-ELBResourceTag -LoadBalancerName my-load-balancer).Tags
```

Output:

Key	Value
---	-----
project	lima
department	digital-media

- Per i dettagli sull'API, vedere [DescribeTags](#) in AWS Tools for PowerShell Cmdlet Reference.

Join-ELBSecurityGroupToLoadBalancer

Il seguente esempio di codice mostra come utilizzare `Join-ELBSecurityGroupToLoadBalancer`

Strumenti per PowerShell

Esempio 1: Questo esempio sostituisce il gruppo di sicurezza corrente per il sistema di bilanciamento del carico specificato con il gruppo di sicurezza specificato.

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -  
SecurityGroup sg-87654321
```

Output:

```
sg-87654321
```

Esempio 2: per mantenere il gruppo di sicurezza corrente e specificare un gruppo di sicurezza aggiuntivo, specificare sia il gruppo di sicurezza esistente che quello nuovo.

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -  
SecurityGroup @"( "sg-12345678", "sg-87654321" )
```

Output:

```
sg-12345678
```

```
sg-87654321
```

- Per i dettagli sull'API, vedere [ApplySecurityGroupsToLoadBalancer](#) in AWS Tools for PowerShell Cmdlet Reference.

New-ELBAppCookieStickinessPolicy

Il seguente esempio di codice mostra come utilizzare `New-ELBAppCookieStickinessPolicy`

Strumenti per PowerShell

Esempio 1: Questo esempio crea una politica di persistenza che segue la durata delle sessioni permanenti del cookie generato dall'applicazione specificato.

```
New-ELBAppCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-app-cookie-policy -CookieName my-app-cookie
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [CreateAppCookieStickinessPolicy](#) AWS Tools for PowerShell

New-ELBLBCookieStickinessPolicy

Il seguente esempio di codice mostra come utilizzare `New-ELBLBCookieStickinessPolicy`

Strumenti per PowerShell

Esempio 1: Questo esempio crea una politica di persistenza con una durata delle sessioni permanenti controllata dal periodo di scadenza specificato (in secondi).

```
New-ELBLBCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-duration-cookie-policy -CookieExpirationPeriod 60
```

Esempio 2: Questo esempio crea una politica di persistenza con una durata delle sessioni permanenti controllata dalla durata del browser (user-agent).

```
New-ELBLBCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-duration-cookie-policy
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [CreateLbCookieStickinessPolicy](#) AWS Tools for PowerShell

New-ELBLoadBalancer

Il seguente esempio di codice mostra come utilizzare New-ELBLoadBalancer

Strumenti per PowerShell

Esempio 1: questo esempio crea un sistema di bilanciamento del carico con un listener HTTP in un VPC.

```
$httpListener = New-Object Amazon.ElasticLoadBalancing.Model.Listener
$httpListener.Protocol = "http"
$httpListener.LoadBalancerPort = 80
$httpListener.InstanceProtocol = "http"
$httpListener.InstancePort = 80
New-ELBLoadBalancer -LoadBalancerName my-vpc-load-balancer -SecurityGroup sg-
a61988c3 -Subnet subnet-15aaab61 -Listener $httpListener

my-vpc-load-balancer-1234567890.us-west-2.elb.amazonaws.com
```

Esempio 2: questo esempio crea un sistema di bilanciamento del carico con un listener HTTP in EC2-Classic.

```
New-ELBLoadBalancer -LoadBalancerName my-classic-load-balancer -AvailabilityZone us-
west-2a -Listener $httpListener
```

Output:

```
my-classic-load-balancer-123456789.us-west-2.elb.amazonaws.com
```

Esempio 3: questo esempio crea un sistema di bilanciamento del carico con un listener HTTPS.

```
$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
```

- Per i dettagli sull'API, vedere [CreateLoadBalancer](#) in AWS Tools for PowerShell Cmdlet Reference.

New-ELBLoadBalancerListener

Il seguente esempio di codice mostra come utilizzare `New-ELBLoadBalancerListener`

Strumenti per PowerShell

Esempio 1: questo esempio aggiunge un listener HTTPS al sistema di bilanciamento del carico specificato.

```
$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
```

- Per i dettagli sull'API, vedere [CreateLoadBalancerListeners](#) in AWS Tools for PowerShell Cmdlet Reference.

New-ELBLoadBalancerPolicy

Il seguente esempio di codice mostra come utilizzare `New-ELBLoadBalancerPolicy`

Strumenti per PowerShell

Esempio 1: Questo esempio crea una nuova politica di protocollo proxy per un sistema di bilanciamento del carico specificato.

```
$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
```

- Per i dettagli sull'API, vedere [CreateLoadBalancerPolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Register-ELBInstanceWithLoadBalancer

Il seguente esempio di codice mostra come utilizzare. Register-ELBInstanceWithLoadBalancer

Strumenti per PowerShell

Esempio 1: questo esempio registra l'istanza EC2 specificata con il sistema di bilanciamento del carico specificato.

```
Register-ELBInstanceWithLoadBalancer -LoadBalancerName my-load-balancer -Instance i-12345678
```

Output:

```
InstanceId
-----
i-12345678
i-87654321
```

- Per i dettagli sull'API, vedere [RegisterInstancesWithLoadBalancer](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-ELBInstanceFromLoadBalancer

Il seguente esempio di codice mostra come utilizzare. Remove-ELBInstanceFromLoadBalancer

Strumenti per PowerShell

Esempio 1: questo esempio rimuove l'istanza EC2 specificata dal sistema di bilanciamento del carico specificato. Prima di procedere con l'operazione, viene richiesta una conferma, a meno che non si specifichi anche il parametro Force.

```
Remove-ELBInstanceFromLoadBalancer -LoadBalancerName my-load-balancer -Instance
i-12345678
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DeregisterInstancesFromLoadBalancer](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-ELBLoadBalancer

Il seguente esempio di codice mostra come utilizzare `Remove-ELBLoadBalancer`

Strumenti per PowerShell

Esempio 1: questo esempio elimina il sistema di bilanciamento del carico specificato. Prima di procedere con l'operazione, viene richiesta una conferma, a meno che non si specifichi anche il parametro `Force`.

```
Remove-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

Output:

```
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"):
```

- Per i dettagli sull'API, vedere [DeleteLoadBalancer](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-ELBLoadBalancerListener

Il seguente esempio di codice mostra come utilizzare. Remove-ELBLoadBalancerListener

Strumenti per PowerShell

Esempio 1: Questo esempio elimina il listener sulla porta 80 per il sistema di bilanciamento del carico specificato. Prima di procedere con l'operazione, viene richiesta una conferma, a meno che non si specifichi anche il parametro Force.

```
Remove-ELBLoadBalancerListener -LoadBalancerName my-load-balancer -LoadBalancerPort 80
```

Output:

```
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"):
```

- Per i dettagli sull'API, vedere [DeleteLoadBalancerListeners](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-ELBLoadBalancerPolicy

Il seguente esempio di codice mostra come utilizzare. Remove-ELBLoadBalancerPolicy

Strumenti per PowerShell

Esempio 1: Questo esempio elimina la politica specificata dal sistema di bilanciamento del carico specificato. Prima di procedere con l'operazione, viene richiesta una conferma, a meno che non si specifichi anche il parametro Force.

```
Remove-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-duration-cookie-policy
```

Output:

```
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"):
```

- Per i dettagli sull'API, vedere [DeleteLoadBalancerPolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-ELBResourceTag

Il seguente esempio di codice mostra come utilizzare. Remove-ELBResourceTag

Strumenti per PowerShell

Esempio 1: Questo esempio rimuove il tag specificato dal sistema di bilanciamento del carico specificato. Prima di procedere con l'operazione, viene richiesta una conferma, a meno che non si specifichi anche il parametro Force. La sintassi utilizzata da questo esempio richiede la PowerShell versione 3 o successiva.

```
Remove-ELBResourceTag -LoadBalancerName my-load-balancer -Tag @{ Key="project" }
```

Output:

```
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"):
```

Esempio 2: con Powershell versione 2, è necessario utilizzare New-Object per creare il tag per il parametro Tag.

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.TagKeyOnly
$tag.Key = "project"
```

```
Remove-ELBResourceTag -Tag $tag -Force
```

- Per i dettagli sull'API, vedere [RemoveTags](#) in AWS Tools for PowerShell Cmdlet Reference.

Set-ELBHealthCheck

Il seguente esempio di codice mostra come utilizzare. Set-ELBHealthCheck

Strumenti per PowerShell

Esempio 1: Questo esempio configura le impostazioni del controllo dello stato del carico per il bilanciamento del carico specificato.

```
Set-ELBHealthCheck -LoadBalancerName my-load-balancer `
>> -HealthCheck_HealthyThreshold 2 `
>> -HealthCheck_UnhealthyThreshold 2 `
>> -HealthCheck_Target "HTTP:80/ping" `
>> -HealthCheck_Interval 30 `
>> -HealthCheck_Timeout 3
```

Output:

```
HealthyThreshold    : 2
Interval            : 30
Target              : HTTP:80/ping
Timeout             : 3
UnhealthyThreshold : 2
```

- Per i dettagli sull'API, vedere [ConfigureHealthCheck](#) in AWS Tools for PowerShell Cmdlet Reference.

Set-ELBLoadBalancerListenerSSLCertificate

Il seguente esempio di codice mostra come utilizzare. Set-ELBLoadBalancerListenerSSLCertificate

Strumenti per PowerShell

Esempio 1: questo esempio sostituisce il certificato che interrompe le connessioni SSL per il listener specificato.

```
Set-ELBLoadBalancerListenerSSLCertificate -LoadBalancerName my-load-balancer `
>> -LoadBalancerPort 443 `
>> -SSLCertificateId "arn:aws:iam::123456789012:server-certificate/new-server-cert"
```

- Per i dettagli sull'API, vedere [SetLoadBalancerListenerSslCertificate](#) in Cmdlet Reference.AWS Tools for PowerShell

Set-ELBLoadBalancerPolicyForBackendServer

Il seguente esempio di codice mostra come utilizzare. Set-ELBLoadBalancerPolicyForBackendServer

Strumenti per PowerShell

Esempio 1: Questo esempio sostituisce le politiche per la porta specificata con la politica specificata.

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -
InstancePort 80 -PolicyName my-ProxyProtocol-policy
```

Esempio 2: Questo esempio rimuove tutte le politiche associate alla porta specificata.

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -
InstancePort 80
```

- Per i dettagli sull'API, vedere [SetLoadBalancerPoliciesForBackendServer](#) in AWS Tools for PowerShell Cmdlet Reference.

Set-ELBLoadBalancerPolicyOfListener

Il seguente esempio di codice mostra come utilizzare. Set-ELBLoadBalancerPolicyOfListener

Strumenti per PowerShell

Esempio 1: Questo esempio sostituisce le politiche per il listener specificato con la politica specificata.

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 443 -PolicyName my-SSLNegotiation-policy
```


Esempio 2: Questo esempio rimuove tutte le politiche associate al listener specificato.

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -  
LoadBalancerPort 443
```

- Per i dettagli sull'API, vedere [SetLoadBalancerPoliciesOfListener](#) in AWS Tools for PowerShell Cmdlet Reference.

Elastic Load Balancing - Esempi della versione 2 che utilizzano Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with Elastic Load Balancing - Versione 2.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Add-ELB2ListenerCertificate

Il seguente esempio di codice mostra come utilizzare `Add-ELB2ListenerCertificate`.

Strumenti per PowerShell

Esempio 1: Questo esempio aggiunge un certificato aggiuntivo al Listener specificato.

```
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'}
```

Output:

```
CertificateArn
IsDefault
-----
-----
arn:aws:acm:us-east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97
False
```

- Per i dettagli sull'API, vedere [AddListenerCertificates](#) in AWS Tools for PowerShell Cmdlet Reference.

Add-ELB2Tag

Il seguente esempio di codice mostra come utilizzare. Add-ELB2Tag

Strumenti per PowerShell

Esempio 1: Questo esempio aggiunge un nuovo tag alla **AWS.Tools.ElasticLoadBalancingV2** risorsa specificata.

```
Add-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Tag @{Key =
'productVersion'; Value = '1.0.0'}
```

- Per i dettagli sull'API, vedere [AddTags](#) in AWS Tools for PowerShell Cmdlet Reference.

Edit-ELB2Listener

Il seguente esempio di codice mostra come utilizzare. Edit-ELB2Listener

Strumenti per PowerShell

Esempio 1: Questo esempio modifica l'azione predefinita dell'ascoltatore specificato in risposta fissa.

```
$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

```

Output:

```

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        :

```

- Per i dettagli sull'API, vedere [ModifyListener](#) in Cmdlet Reference.AWS Tools for PowerShell

Edit-ELB2LoadBalancerAttribute

Il seguente esempio di codice mostra come utilizzare `Edit-ELB2LoadBalancerAttribute`

Strumenti per PowerShell

Esempio 1: Questo esempio modifica gli attributi del sistema di bilanciamento del carico specificato.

```

Edit-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Attribute @{Key =
'deletion_protection.enabled'; Value = 'true'}

```

Output:

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

```

- Per i dettagli sull'API, vedere [ModifyLoadBalancerAttributes](#) in AWS Tools for PowerShell Cmdlet Reference.

Edit-ELB2Rule

Il seguente esempio di codice mostra come utilizzare. Edit-ELB2Rule

Strumenti per PowerShell

Esempio 1: Questo esempio modifica le configurazioni delle regole Listener specificate.

```

$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

```

Output:

```

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

```

- Per i dettagli sull'API, vedere [ModifyRule](#) in AWS Tools for PowerShell Cmdlet Reference.

Edit-ELB2TargetGroup

Il seguente esempio di codice mostra come utilizzare. Edit-ELB2TargetGroup

Strumenti per PowerShell

Esempio 1: questo esempio modifica le proprietà del gruppo di destinazione specificato.

```
Edit-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -HealthCheckIntervalSecond 60 -HealthCheckPath '/index.html' -HealthCheckPort 8080
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [ModifyTargetGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

Edit-ELB2TargetGroupAttribute

Il seguente esempio di codice mostra come utilizzare. Edit-ELB2TargetGroupAttribute

Strumenti per PowerShell

Esempio 1: questo esempio modifica l'attributo deregistration_delay del gruppo target specificato.

```
Edit-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Attribute @{Key = 'deregistration_delay.timeout_seconds'; Value = 600}
```

Output:

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

- Per i dettagli sull'API, vedere in Cmdlet Reference. [ModifyTargetGroupAttributes](#) AWS Tools for PowerShell

Get-ELB2AccountLimit

Il seguente esempio di codice mostra come utilizzare. `Get-ELB2AccountLimit`

Strumenti per PowerShell

Esempio 1: Questo comando elenca i limiti degli account ELB2 per una determinata regione.

```
Get-ELB2AccountLimit
```

Output:

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

```

- Per i dettagli sull'API, vedere [DescribeAccountLimits](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ELB2Listener

Il seguente esempio di codice mostra come utilizzare. `Get-ELB2Listener`

Strumenti per PowerShell

Esempio 1: questo esempio descrive gli ascoltatori dell'ALB/NLB specificato.

```

Get-ELB2Listener -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'

```

Output:

```

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

```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [DescribeListeners](#) AWS Tools for PowerShell

Get-ELB2ListenerCertificate

Il seguente esempio di codice mostra come utilizzare. `Get-ELB2ListenerCertificate`

Strumenti per PowerShell

Esempio 1: questo esempio descrive il certificato per l'ascoltatore specificato.

```
Get-ELB2ListenerCertificate -ListenerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Output:

```
CertificateArn
IsDefault
-----
-----
arn:aws:acm:us-east-1:123456789012:certificate/5fc7c092-68bf-4862-969c-22fd48b6e17c
True
```

- Per i dettagli sull'API, vedere [DescribeListenerCertificates](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ELB2LoadBalancer

Il seguente esempio di codice mostra come utilizzare. `Get-ELB2LoadBalancer`

Strumenti per PowerShell

Esempio 1: questo esempio mostra tutti i sistemi di bilanciamento del carico per una determinata regione.

```
Get-ELB2LoadBalancer
```

Output:

```
AvailabilityZones : {us-east-1c}
```



```

CanonicalHostedZoneId : Z26RNL4JYFT0TI
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

```

- Per i dettagli sull'API, vedere [DescribeLoadBalancers](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ELB2LoadBalancerAttribute

Il seguente esempio di codice mostra come utilizzare `Get-ELB2LoadBalancerAttribute`

Strumenti per PowerShell

Esempio 1: questo comando descrive gli attributi di un determinato Load Balancer.

```

Get-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/net/test-elb/238d34ad8d94bc2e'

```

Output:

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	

- Per i dettagli sull'API, vedere [DescribeLoadBalancerAttributes](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ELB2Rule

Il seguente esempio di codice mostra come utilizzare. Get-ELB2Rule

Strumenti per PowerShell

Esempio 1: questo esempio descrive le regole del listener per l'ARN del listener specificato.

```
Get-ELB2Rule -ListenerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DescribeRules](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ELB2SSLPolicy

Il seguente esempio di codice mostra come utilizzare. Get-ELB2SSLPolicy

Strumenti per PowerShell

Esempio 1: questo esempio elenca tutte le politiche di ascolto disponibili per la ElasticLoadBalancing versione 2.

```
Get-ELB2SSLPolicy
```

Output:

```
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}
```

- Per i dettagli sull'API, vedere [DescribeSslPolicies](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ELB2Tag

Il seguente esempio di codice mostra come utilizzare. Get-ELB2Tag

Strumenti per PowerShell

Esempio 1: questo esempio elenca i tag per la risorsa specificata.

```
Get-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Output:

```
ResourceArn
           Tags
-----
-----
arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f {stage, internalName, version}
```

- Per i dettagli sull'API, vedere [DescribeTags](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ELB2TargetGroup

Il seguente esempio di codice mostra come utilizzare. Get-ELB2TargetGroup

Strumenti per PowerShell

Esempio 1: questo esempio descrive il gruppo target specificato.

```
Get-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Output:

```
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

```

- Per i dettagli sull'API, vedere [DescribeTargetGroups](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ELB2TargetGroupAttribute

Il seguente esempio di codice mostra come utilizzare `Get-ELB2TargetGroupAttribute`

Strumenti per PowerShell

Esempio 1: questo esempio descrive gli attributi del gruppo target specificato.

```

Get-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'

```

Output:

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

- Per i dettagli sull'API, vedere [DescribeTargetGroupAttributes](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ELB2TargetHealth

Il seguente esempio di codice mostra come utilizzare. `Get-ELB2TargetHealth`

Strumenti per PowerShell

Esempio 1: Questo esempio restituisce lo stato di salute dei Target presenti nel Target Group specificato.

```
Get-ELB2TargetHealth -TargetGroupArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Output:

HealthCheckPort	Target	TargetHealth
80	Amazon.ElasticLoadBalancingV2.Model.TargetDescription	Amazon.ElasticLoadBalancingV2.Model.TargetHealth

- Per i dettagli sull'API, vedere [DescribeTargetHealth](#) in AWS Tools for PowerShell Cmdlet Reference.

New-ELB2Listener

Il seguente esempio di codice mostra come utilizzare. `New-ELB2Listener`

Strumenti per PowerShell

Esempio 1: Questo esempio crea un nuovo listener ALB con l'azione predefinita «Forward» per inviare il traffico a un gruppo di destinazione specificato.

```
$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

```

Output:

```

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        :

```

- Per i dettagli sull'API, vedere [CreateListener](#) in AWS Tools for PowerShell Cmdlet Reference.

New-ELB2LoadBalancer

Il seguente esempio di codice mostra come utilizzare `New-ELB2LoadBalancer`

Strumenti per PowerShell

Esempio 1: Questo esempio crea un nuovo sistema di bilanciamento del carico delle applicazioni rivolto a Internet con due sottoreti.

```

New-ELB2LoadBalancer -Type application -Scheme internet-facing -IpAddressType
ipv4 -Name 'New-Test-ALB' -SecurityGroup 'sg-07c3414abb8811cbd' -subnet 'subnet-
c37a67a6', 'subnet-fc02eea0'

```

Output:

```

AvailabilityZones : {us-east-1b, us-east-1a}
CanonicalHostedZoneId : Z35SXD0TRQ7X7K
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

```

- Per i dettagli sull'API, vedere [CreateLoadBalancer](#) in AWS Tools for PowerShell Cmdlet Reference.

New-ELB2Rule

Il seguente esempio di codice mostra come utilizzare. `New-ELB2Rule`

Strumenti per PowerShell

Esempio 1: Questo esempio crea una nuova regola Listener con un'azione a risposta fissa basata sul valore dell'intestazione del cliente per il Listener specificato.

```

$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

```


Output:

```

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

```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [CreateRule](#) AWS Tools for PowerShell

New-ELB2TargetGroup

Il seguente esempio di codice mostra come utilizzare `New-ELB2TargetGroup`

Strumenti per PowerShell

Esempio 1: Questo esempio crea un nuovo gruppo Target con i parametri forniti.

```

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'

```

Output:

```

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
```

- Per i dettagli sull'API, vedere [CreateTargetGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

Register-ELB2Target

Il seguente esempio di codice mostra come utilizzare. Register-ELB2Target

Strumenti per PowerShell

Esempio 1: questo esempio registra l'istanza 'i-0672a4c4cdeae3111' con il gruppo target specificato.

```
Register-ELB2Target -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Target @{Port = 80; Id =
'i-0672a4c4cdeae3111'}
```

- Per [RegisterTargets](#) i dettagli sull'API, vedere in Cmdlet AWS Tools for PowerShell Reference.

Remove-ELB2Listener

Il seguente esempio di codice mostra come utilizzare. Remove-ELB2Listener

Strumenti per PowerShell

Esempio 1: Questo esempio elimina il Listener specificato.

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Output:

```
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
```

Esempio 2: Questo esempio rimuove il listener specificato dal Load balancer.

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618'
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DeleteListener](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-ELB2ListenerCertificate

Il seguente esempio di codice mostra come utilizzare `Remove-ELB2ListenerCertificate`

Strumenti per PowerShell

Esempio 1: questo esempio rimuove il certificato specificato dal gruppo Target specificato.

```
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'
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [RemoveListenerCertificates](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-ELB2LoadBalancer

Il seguente esempio di codice mostra come utilizzare. Remove-ELB2LoadBalancer

Strumenti per PowerShell

Esempio 1: questo esempio elimina il Load balancer specificato.

```
Remove-ELB2LoadBalancer -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DeleteLoadBalancer](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-ELB2Rule

Il seguente esempio di codice mostra come utilizzare. Remove-ELB2Rule

Strumenti per PowerShell

Esempio 1: questo esempio rimuove la regola specificata dal Listener

```
Remove-ELB2Rule -RuleArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-alb/3651b4394dd9a24f/3873f123b98f7618/4b25eb10a42e33ab'
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DeleteRule](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-ELB2Tag

Il seguente esempio di codice mostra come utilizzare. Remove-ELB2Tag

Strumenti per PowerShell

Esempio 1: Questo esempio rimuove il tag per la chiave specificata.

```
Remove-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -TagKey
'productVersion'
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [RemoveTags](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-ELB2TargetGroup

Il seguente esempio di codice mostra come utilizzare. Remove-ELB2TargetGroup

Strumenti per PowerShell

Esempio 1: questo esempio rimuove il gruppo target specificato.

```
Remove-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/testsssss/4e0b6076bc6483a7'
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DeleteTargetGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

Set-ELB2IpAddressType

Il seguente esempio di codice mostra come utilizzare. Set-ELB2IpAddressType

Strumenti per PowerShell

Esempio 1: Questo esempio modifica il tipo di indirizzo IP del Load Balancer da 'IPv4' a "DualStack

```
Set-ELB2IpAddressType -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -IpAddressType dualstack
```

Output:

```
Value
-----
dualstack
```

- Per i dettagli sull'API, vedere [SetIpAddressType](#) in AWS Tools for PowerShell Cmdlet Reference.

Set-ELB2RulePriority

Il seguente esempio di codice mostra come utilizzare. Set-ELB2RulePriority

Strumenti per PowerShell

Esempio 1: Questo esempio modifica la priorità della regola del listener specificata.

```
Set-ELB2RulePriority -RulePriority -RulePriority @{Priority = 11; RuleArn =  
'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-  
alb/3651b4394dd9a24f/a4eb199fa5046f80/dbf4c6dcef3ec6f8'}
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [SetRulePriorities](#) in AWS Tools for PowerShell Cmdlet Reference.

Set-ELB2SecurityGroup

Il seguente esempio di codice mostra come utilizzare. Set-ELB2SecurityGroup

Strumenti per PowerShell

Esempio 1: Questo esempio aggiunge il gruppo di sicurezza 'sg-07c3414abb8811cbd' al Load balancer specificato.

```
Set-ELB2SecurityGroup -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-  
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -SecurityGroup  
'sg-07c3414abb8811cbd'
```

Output:

```
sg-07c3414abb8811cbd
```

- Per i [SetSecurityGroups](#) dettagli AWS Tools for PowerShell sull'API, vedere in Cmdlet Reference.

Set-ELB2Subnet

Il seguente esempio di codice mostra come utilizzare. Set-ELB2Subnet

Strumenti per PowerShell

Esempio 1: questo esempio modifica le sottoreti del Load balancer specificato.

```
Set-ELB2Subnet -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Subnet 'subnet-7d8a0a51', 'subnet-c37a67a6'
```

Output:

LoadBalancerAddresses	SubnetId	ZoneName
{}	subnet-7d8a0a51	us-east-1c
{}	subnet-c37a67a6	us-east-1b

- Per i dettagli sull'API, vedere [SetSubnets](#) in Cmdlet Reference.AWS Tools for PowerShell

Unregister-ELB2Target

Il seguente esempio di codice mostra come utilizzare. Unregister-ELB2Target

Strumenti per PowerShell

Esempio 1: questo esempio annulla la registrazione dell'istanza 'i-0672a4c4cdeae3111' dal gruppo Target specificato.

```
$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'
```


- Per [DeregisterTargets](#) dettagli sull'API AWS Tools for PowerShell , vedere in Cmdlet Reference.

Esempi di Amazon FSx con Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon FSx. AWS Tools for PowerShell

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Add-FSXResourceTag

Il seguente esempio di codice mostra come utilizzare Add-FSXResourceTag.

Strumenti per PowerShell

Esempio 1: Questo esempio aggiunge tag alla risorsa data.

```
Add-FSXResourceTag -ResourceARN "arn:aws:fsx:eu-west-1:123456789012:file-system/fs-01cd23bc4bdf5678a" -Tag @{Key="Users";Value="Test"} -PassThru
```

Output:

```
arn:aws:fsx:eu-west-1:123456789012:file-system/fs-01cd23bc4bdf5678a
```

- Per i dettagli sull'API, vedere [TagResource](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-FSXBackup

Il seguente esempio di codice mostra come utilizzare. Get-FSXBackup

Strumenti per PowerShell

Esempio 1: Questo esempio recupera i backup creati da ieri per l'ID del file system specificato.

```
Get-FSXBackup -Filter @{Name="file-system-id";Values=$fsx.FileSystemId} | Where-Object CreationTime -gt (Get-Date).AddDays(-1)
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DescribeBackups](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-FSXFileSystem

Il seguente esempio di codice mostra come utilizzare. Get-FSXFileSystem

Strumenti per PowerShell

Esempio 1: Questo esempio restituisce la descrizione di un determinato FileSystemID.

```
Get-FSXFileSystem -FileSystemId fs-01cd23bc4bdf5678a
```

Output:

```
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

```

- Per i dettagli sull'API, vedere [DescribeFileSystems](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-FSXResourceTagList

Il seguente esempio di codice mostra come utilizzare. Get-FSXResourceTagList

Strumenti per PowerShell

Esempio 1: Questo esempio elenca i tag per la risorsa arn fornita.

```
Get-FSXResourceTagList -ResourceARN $fsx.ResourceARN
```

Output:

Key	Value
---	-----
FSx-Service	Windows
Users	Dev

- Per i dettagli sull'API, vedere [ListTagsForResource](#) in AWS Tools for PowerShell Cmdlet Reference.

New-FSXBackup

Il seguente esempio di codice mostra come utilizzare. New-FSXBackup

Strumenti per PowerShell

Esempio 1: Questo esempio crea un backup del file system specificato.

```
New-FSXBackup -FileSystemId fs-0b1fac2345623456ba
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [CreateBackup](#) in AWS Tools for PowerShell Cmdlet Reference.

New-FSXFileSystem

Il seguente esempio di codice mostra come utilizzare. New-FSXFileSystem

Strumenti per PowerShell

Esempio 1: Questo esempio crea un nuovo file system Windows da 300 GB, che consente l'accesso dalla sottorete specificata, che supporta una velocità di trasmissione fino a 8 megabyte al secondo. Il nuovo file system viene aggiunto automaticamente alla Microsoft Active Directory specificata.

```
New-FSXFileSystem -FileSystemType WINDOWS -StorageCapacity
300 -SubnetId subnet-1a2b3c4d5e6f -WindowsConfiguration
@{ThroughputCapacity=8;ActiveDirectoryId='d-1a2b3c4d'}
```

Output:

```

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

```

- Per i dettagli sull'API, vedere [CreateFileSystem](#) in AWS Tools for PowerShell Cmdlet Reference.

New-FSXFileSystemFromBackup

Il seguente esempio di codice mostra come utilizzare `New-FSXFileSystemFromBackup`

Strumenti per PowerShell

Esempio 1: questo esempio crea un nuovo file system Amazon FSx da un backup esistente di Amazon FSx for Windows File Server.

```

New-FSXFileSystemFromBackup -BackupId $backupID -Tag @{Key="tag:Name";Value="from-
manual-backup"} -SubnetId $SubnetID -SecurityGroupId $SG_ID -WindowsConfiguration
@{ThroughputCapacity=8;ActiveDirectoryId=$DirectoryID}

```

Output:

```

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

```

- Per i dettagli sull'API, vedere [CreateFileSystemFromBackup](#) in Cmdlet Reference.AWS Tools for PowerShell

Remove-FSXBackup

Il seguente esempio di codice mostra come utilizzare. Remove-FSXBackup

Strumenti per PowerShell

Esempio 1: Questo esempio rimuove l'id di backup specificato.

```
Remove-FSXBackup -BackupId $backupID
```

Output:

```

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

```

- Per i dettagli sull'API, vedere [DeleteBackup](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-FSXFileSystem

Il seguente esempio di codice mostra come utilizzare. Remove-FSXFileSystem

Strumenti per PowerShell

Esempio 1: Questo esempio rimuove l'ID del file system FSX specificato.

```
Remove-FSXFileSystem -FileSystemId fs-012ff34e567890120
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DeleteFileSystem](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-FSXResourceTag

Il seguente esempio di codice mostra come utilizzare. Remove-FSXResourceTag

Strumenti per PowerShell

Esempio 1: Questo esempio rimuove il tag di risorsa per la risorsa ARN del file system FSX specificata.

```
Remove-FSXResourceTag -ResourceARN $FSX.ResourceARN -TagKey Users
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [UntagResource](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-FSXFileSystem

Il seguente esempio di codice mostra come utilizzare. Update-FSXFileSystem

Strumenti per PowerShell

Esempio 1: Questo esempio aggiorna i giorni di conservazione automatica dei backup del file system FSX tramite UpdateFileSystemWindowsConfiguration.

```
$UpdateFSXWinConfig = [Amazon.FSx.Model.UpdateFileSystemWindowsConfiguration]::new()
$UpdateFSXWinConfig.AutomaticBackupRetentionDays = 35
Update-FSXFileSystem -FileSystemId $FSX.FileSystemId -WindowsConfiguration
$UpdateFSXWinConfig
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [UpdateFileSystem](#) in AWS Tools for PowerShell Cmdlet Reference.

AWS Glue esempi di utilizzo di Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with AWS Glue.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

New-GLUEJob

Il seguente esempio di codice mostra come utilizzare New-GLUEJob.

Strumenti per PowerShell

Esempio 1: Questo esempio crea un nuovo lavoro in AWS Glue. Il valore del nome del comando è sempre **glueet1**. AWS Glue supporta l'esecuzione di script di lavoro scritti in Python o Scala. In questo esempio, lo script di lavoro (MyTestGlueJob.py) è scritto in Python. I parametri Python vengono specificati nella **\$DefArgs** variabile e quindi passati al PowerShell comando nel **DefaultArguments** parametro, che accetta una tabella hash. I parametri della **\$JobParams** variabile provengono dall' CreateJob API, documentata nell'argomento Jobs (<https://docs.aws.amazon.com/glue/latest/dg/aws-glue-api-jobs-job.html>) del riferimento all'API AWS Glue.

```
$Command = New-Object Amazon.Glue.Model.JobCommand
$Command.Name = 'glueet1'
$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
```

- Per i dettagli sull'API, vedere [CreateJob](#) in AWS Tools for PowerShell Cmdlet Reference.

AWS Health esempi di utilizzo di Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with AWS Health.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Get-HLTHEvent

Il seguente esempio di codice mostra come utilizzare `Get-HLTHEvent`.

Strumenti per PowerShell

Esempio 1: questo comando restituisce gli eventi da AWS Personal Health Dashboard. L'utente aggiunge il parametro `-Region` per visualizzare gli eventi disponibili per il servizio nella regione Stati Uniti orientali (Virginia settentrionale), ma il parametro `-Filter_Region` filtra gli eventi registrati nelle regioni UE (Londra) e Stati Uniti occidentali (Oregon) (`eu-west-2` e `us-west-2`). Il `StartTime` parametro `-Filter_` filtra per un intervallo di volte in cui gli eventi possono iniziare, mentre il `EndTime` parametro `-Filter_` filtra per un intervallo di volte in cui gli eventi possono terminare. Il risultato è un evento di manutenzione pianificato per RDS che inizia all'interno dell'intervallo `-Filter_` specificato e termina nell'`StartTime` intervallo pianificato `-Filter_`. `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"}
```

Output:

```
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
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [DescribeEvents](#) AWS Tools for PowerShell

Esempi IAM che utilizzano Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with IAM.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Add-IAMClientIDToOpenIDConnectProvider

Il seguente esempio di codice mostra come utilizzare `Add-IAMClientIDToOpenIDConnectProvider`.

Strumenti per PowerShell

Esempio 1: questo comando aggiunge l'ID client (o il pubblico) **my-application-ID** al provider OIDC esistente denominato **server.example.com**

```
Add-IAMClientIDToOpenIDConnectProvider -ClientID "my-application-ID"
-OpenIDConnectProviderARN "arn:aws:iam::123456789012:oidc-provider/
server.example.com"
```

- Per i dettagli sull'API, vedere [AddClientIdToOpenIdConnectProvider](#) in AWS Tools for PowerShell Cmdlet Reference.

Add-IAMRoleTag

Il seguente esempio di codice mostra come utilizzare. Add-IAMRoleTag

Strumenti per PowerShell

Esempio 1: questo esempio aggiunge un tag a Role in Identity Management Service

```
Add-IAMRoleTag -RoleName AdminRoleaccess -Tag @{ Key = 'abac'; Value = 'testing'}
```

- Per i dettagli sull'API, vedere [TagRole](#) in AWS Tools for PowerShell Cmdlet Reference.

Add-IAMRoleToInstanceProfile

Il seguente esempio di codice mostra come utilizzare. Add-IAMRoleToInstanceProfile

Strumenti per PowerShell

Esempio 1: Questo comando aggiunge il ruolo denominato **S3Access** a un profilo di istanza esistente denominato **webserver**. Per creare il profilo dell'istanza, utilizzate il **New-IAMInstanceProfile** comando. Dopo aver creato il profilo dell'istanza e averlo associato a un ruolo utilizzando questo comando, puoi collegarlo a un'istanza EC2. A tale scopo, utilizza il **New-EC2Instance** cmdlet con il parametro **InstanceProfile_Arn** o con il **InstanceProfile-Name** parametro per avviare la nuova istanza.

```
Add-IAMRoleToInstanceProfile -RoleName "S3Access" -InstanceProfileName "webserver"
```

- Per i dettagli sull'API, vedere [AddRoleToInstanceProfile](#) in AWS Tools for PowerShell Cmdlet Reference.

Add-IAMUserTag

Il seguente esempio di codice mostra come utilizzare. Add-IAMUserTag

Strumenti per PowerShell

Esempio 1: questo esempio aggiunge un tag a User in Identity Management Service

```
Add-IAMUserTag -UserName joe -Tag @{ Key = 'abac'; Value = 'testing'}
```

- Per i dettagli sull'API, vedere [TagUser](#) in AWS Tools for PowerShell Cmdlet Reference.

Add-IAMUserToGroup

Il seguente esempio di codice mostra come utilizzare. Add-IAMUserToGroup

Strumenti per PowerShell

Esempio 1: Questo comando aggiunge l'utente denominato **Bob** al gruppo denominato **Admins**.

```
Add-IAMUserToGroup -UserName "Bob" -GroupName "Admins"
```

- Per i dettagli sull'API, vedere [AddUserToGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

Disable-IAMMFADevice

Il seguente esempio di codice mostra come utilizzare. Disable-IAMMFADevice

Strumenti per PowerShell

Esempio 1: questo comando disabilita il dispositivo hardware MFA associato all'**Bob**utente con il numero di serie. **123456789012**

```
Disable-IAMMFADevice -UserName "Bob" -SerialNumber "123456789012"
```

Esempio 2: questo comando disabilita il dispositivo MFA virtuale associato all'**David**utente che dispone dell'ARN. **arn:aws:iam::210987654321:mfa/David** Tieni presente che il dispositivo MFA virtuale non viene eliminato dall'account. Il dispositivo virtuale è ancora presente e appare nell'output del **Get-IAMVirtualMFADevice** comando. Prima di poter creare un nuovo dispositivo MFA virtuale per lo stesso utente, è necessario eliminare quello precedente utilizzando il **Remove-IAMVirtualMFADevice** comando.

```
Disable-IAMMFADevice -UserName "David" -SerialNumber "arn:aws:iam::210987654321:mfa/David"
```

- Per i dettagli sull'API, vedere [DeactivateMfaDevice](#) in AWS Tools for PowerShell Cmdlet Reference.

Edit-IAMPassword

Il seguente esempio di codice mostra come utilizzare. `Edit-IAMPassword`

Strumenti per PowerShell

Esempio 1: Questo comando modifica la password dell'utente che esegue il comando. Questo comando può essere chiamato solo dagli utenti IAM. Se questo comando viene chiamato quando si accede con le credenziali AWS dell'account (root), restituisce un errore. **InvalidUserType**

```
Edit-IAMPassword -OldPassword "MyOldP@ssw0rd" -NewPassword "MyNewP@ssw0rd"
```

- Per i dettagli sull'API, vedere [ChangePassword](#) in Cmdlet Reference.AWS Tools for PowerShell

Enable-IAMMFADevice

Il seguente esempio di codice mostra come utilizzare. `Enable-IAMMFADevice`

Strumenti per PowerShell

Esempio 1: questo comando abilita il dispositivo hardware MFA con il numero di serie **987654321098** e lo associa all'utente. **Bob** Include i primi due codici in sequenza provenienti dal dispositivo.

```
Enable-IAMMFADevice -UserName "Bob" -SerialNumber "987654321098" -  
AuthenticationCode1 "12345678" -AuthenticationCode2 "87654321"
```

Esempio 2: Questo esempio crea e abilita un dispositivo MFA virtuale. Il primo comando crea il dispositivo virtuale e restituisce la rappresentazione dell'oggetto del dispositivo nella variabile **\$MFADevice**. È possibile utilizzare le **QRCodePng** proprietà **.Base32StringSeed** o per configurare l'applicazione software dell'utente. Il comando finale assegna il dispositivo all'utente **David**, identificandolo in base al numero di serie. Il comando sincronizza inoltre il dispositivo con AWS l'inclusione dei primi due codici in sequenza dal dispositivo MFA virtuale.

```
$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 $MFADevice.SerialNumber  
-SerialNumber $MFADevice.SerialNumber -AuthenticationCode1 "24681357" -AuthenticationCode2  
"13572468"
```

- Per i dettagli sull'API, vedere [EnableMfaDevice](#) in AWS Tools for PowerShell Cmdlet Reference.

Get - IAMAccessKey

Il seguente esempio di codice mostra come utilizzare. Get - IAMAccessKey

Strumenti per PowerShell

Esempio 1: questo comando elenca le chiavi di accesso per l'utente IAM denominato **Bob**. Tieni presente che non puoi elencare le chiavi di accesso segrete per gli utenti IAM. Se le chiavi di accesso segrete vengono perse, è necessario creare nuove chiavi di accesso con il **New - IAMAccessKey** cmdlet.

```
Get-IAMAccessKey -UserName "Bob"
```

Output:

AccessKeyId	CreateDate	Status	UserName
-----	-----	-----	-----
AKIAIOSFODNN7EXAMPLE	12/3/2014 10:53:41 AM	Active	Bob
AKIAI44QH8DHBEXAMPLE	6/6/2013 8:42:26 PM	Inactive	Bob

- Per i dettagli sull'API, vedere [ListAccessKeys](#) in AWS Tools for PowerShell Cmdlet Reference.

Get - IAMAccessKeyLastUsed

Il seguente esempio di codice mostra come utilizzare. Get - IAMAccessKeyLastUsed

Strumenti per PowerShell

Esempio 1: restituisce il nome utente proprietario e le informazioni sull'ultimo utilizzo della chiave di accesso fornita.

```
Get-IAMAccessKeyLastUsed -AccessKeyId ABCDEXAMPLE
```

- Per i dettagli sull'API, vedere [GetAccessKeyLastUsed](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMAccountAlias

Il seguente esempio di codice mostra come utilizzare. `Get-IAMAccountAlias`

Strumenti per PowerShell

Esempio 1: questo comando restituisce l'alias dell'account per. Account AWS

```
Get-IAMAccountAlias
```

Output:

```
ExampleCo
```

- Per i dettagli sull'API, vedere [ListAccountAliases](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMAccountAuthorizationDetail

Il seguente esempio di codice mostra come utilizzare. `Get-IAMAccountAuthorizationDetail`

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene i dettagli di autorizzazione sulle identità nell' AWS account e visualizza l'elenco degli elementi dell'oggetto restituito, inclusi utenti, gruppi e ruoli. Ad esempio, la **UserDetailList** proprietà visualizza i dettagli sugli utenti. Informazioni simili sono disponibili nelle **GroupDetailList** proprietà **RoleDetailList** e.

```
$Details=Get-IAMAccountAuthorizationDetail  
$Details
```

Output:

```
GroupDetailList : {Administrators, Developers, Testers, Backup}  
IsTruncated     : False  
Marker          :  
RoleDetailList  : {TestRole1, AdminRole, TesterRole, clirole...}  
UserDetailList  : {Administrator, Bob, BackupToS3, }
```

```
$Details.UserDetailList
```

Output:

```
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}
```

- Per i dettagli sull'API, vedere [GetAccountAuthorizationDetails](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMAccountPasswordPolicy

Il seguente esempio di codice mostra come utilizzare `Get-IAMAccountPasswordPolicy`

Strumenti per PowerShell

Esempio 1: questo esempio restituisce dettagli sulla politica delle password per l'account corrente. Se non è definita alcuna politica in materia di password per l'account, il comando restituisce un **NoSuchEntity** errore.

```
Get-IAMAccountPasswordPolicy
```

Output:

```
AllowUsersToChangePassword : True
ExpirePasswords             : True
HardExpiry                  : False
MaxPasswordAge              : 90
MinimumPasswordLength      : 8
PasswordReusePrevention    : 20
RequireLowercaseCharacters : True
RequireNumbers              : True
RequireSymbols              : False
RequireUppercaseCharacters : True
```

- Per i dettagli sull'API, vedere [GetAccountPasswordPolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMAccountSummary

Il seguente esempio di codice mostra come utilizzare. `Get-IAMAccountSummary`

Strumenti per PowerShell

Esempio 1: Questo esempio restituisce informazioni sull'utilizzo corrente delle entità IAM e sulle quote correnti delle entità IAM in Account AWS

```
Get-IAMAccountSummary
```

Output:

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
...

```

- Per i dettagli sull'API, vedere [GetAccountSummary](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMAttachedGroupPolicyList

Il seguente esempio di codice mostra come utilizzare `Get-IAMAttachedGroupPolicyList`

Strumenti per PowerShell

Esempio 1: questo comando restituisce i nomi e gli ARN delle politiche gestite allegate al gruppo IAM denominato **Admins** nell' AWS account. Per visualizzare l'elenco delle politiche in linea incorporate nel gruppo, usa il **Get-IAMGroupPolicyList** comando.

```
Get-IAMAttachedGroupPolicyList -GroupName "Admins"
```

Output:

```

PolicyArn                               PolicyName
-----

```

arn:aws:iam::aws:policy/SecurityAudit	SecurityAudit
arn:aws:iam::aws:policy/AdministratorAccess	AdministratorAccess

- Per i dettagli sull'API, vedere [ListAttachedGroupPolicies](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMAttachedRolePolicyList

Il seguente esempio di codice mostra come utilizzare. `Get-IAMAttachedRolePolicyList`

Strumenti per PowerShell

Esempio 1: questo comando restituisce i nomi e gli ARN delle politiche gestite allegate al ruolo IAM denominato **SecurityAuditRole** nell' AWS account. Per visualizzare l'elenco delle politiche in linea incorporate nel ruolo, usa il **Get-IAMRolePolicyList** comando.

```
Get-IAMAttachedRolePolicyList -RoleName "SecurityAuditRole"
```

Output:

PolicyArn	PolicyName
-----	-----
arn:aws:iam::aws:policy/SecurityAudit	SecurityAudit

- Per i dettagli sull'API, vedere [ListAttachedRolePolicies](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMAttachedUserPolicyList

Il seguente esempio di codice mostra come utilizzare. `Get-IAMAttachedUserPolicyList`

Strumenti per PowerShell

Esempio 1: questo comando restituisce i nomi e gli ARN delle politiche gestite per l'utente IAM indicato **Bob** nell' AWS account. Per visualizzare l'elenco delle politiche in linea incorporate nell'utente IAM, usa il **Get-IAMUserPolicyList** comando.

```
Get-IAMAttachedUserPolicyList -UserName "Bob"
```

Output:

PolicyArn	PolicyName
-----	-----
arn:aws:iam::aws:policy/TesterPolicy	TesterPolicy

- Per i dettagli sull'API, vedere [ListAttachedUserPolicies](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMContextKeysForCustomPolicy

Il seguente esempio di codice mostra come utilizzare. `Get-IAMContextKeysForCustomPolicy`

Strumenti per PowerShell

Esempio 1: Questo esempio recupera tutte le chiavi di contesto presenti nella policy JSON fornita. Per fornire più politiche è possibile fornire un elenco di valori separati da virgole.

```
$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
```

- Per i dettagli sull'API, vedere [GetContextKeysForCustomPolicy](#) in Cmdlet Reference. AWS Tools for PowerShell

Get-IAMContextKeysForPrincipalPolicy

Il seguente esempio di codice mostra come utilizzare. `Get-IAMContextKeysForPrincipalPolicy`

Strumenti per PowerShell

Esempio 1: Questo esempio recupera tutte le chiavi di contesto presenti nella policy json fornita e le politiche allegate all'entità IAM (user/role ecc.). Per: `PolicyInputList` puoi fornire più elenchi di valori come valori separati da virgole.

```
$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
```

- Per i dettagli sull'API, vedere [GetContextKeysForPrincipalPolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMCredentialReport

Il seguente esempio di codice mostra come utilizzare `Get-IAMCredentialReport`

Strumenti per PowerShell

Esempio 1: Questo esempio apre il report restituito e lo invia alla pipeline come matrice di righe di testo. La prima riga è l'intestazione con i nomi delle colonne separati da virgole. Ogni riga successiva è la riga di dettaglio per un utente, con ogni campo separato da virgole. Prima di poter visualizzare il report, è necessario generarlo con il **Request-IAMCredentialReport** cmdlet. Per recuperare il report come singola stringa, utilizzare **-Raw** invece di **-AsTextArray**. L'alias **-SplitLines** è accettato anche per lo **-AsTextArray** switch. Per l'elenco completo delle colonne dell'output, consulta il riferimento all'API del servizio. Nota che se non utilizzi **-AsTextArray** or **-SplitLines**, devi estrarre il testo dalla **.Content** proprietà utilizzando la **StreamReader** classe.NET.

```
Request-IAMCredentialReport
```

Output:

```
Description                               State
-----
No report exists. Starting a new report generation task    STARTED
```

```
Get-IAMCredentialReport -AsTextArray
```

Output:

```

user,arn,user_creation_time,password_enabled,password_last_used,password_last_changed,password
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

```

- Per i dettagli sull'API, vedere [GetCredentialReport](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMEntitiesForPolicy

Il seguente esempio di codice mostra come utilizzare. `Get-IAMEntitiesForPolicy`

Strumenti per PowerShell

Esempio 1: questo esempio restituisce un elenco di gruppi, ruoli e utenti IAM a cui è associata la policy **arn:aws:iam::123456789012:policy/TestPolicy**.

```
Get-IAMEntitiesForPolicy -PolicyArn "arn:aws:iam::123456789012:policy/TestPolicy"
```

Output:

```

IsTruncated   : False
Marker        :
PolicyGroups  : {}
PolicyRoles   : {testRole}
PolicyUsers   : {Bob, Theresa}

```

- Per i dettagli sull'API, vedere [ListEntitiesForPolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMGroup

Il seguente esempio di codice mostra come utilizzare. `Get-IAMGroup`

Strumenti per PowerShell

Esempio 1: questo esempio restituisce dettagli sul gruppo **IAMTesters**, inclusa una raccolta di tutti gli utenti IAM che appartengono al gruppo.

```
$results = Get-IAMGroup -GroupName "Testers"
$results
```

Output:

Group	IsTruncated	Marker
Users		
-----	-----	-----

Amazon.IdentityManagement.Model.Group {Theresa, David}	False	

```
$results.Group
```

Output:

```
Arn      : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId   : 3RHNZZGQJ7QHMAEXAMPLE1
GroupName : Testers
Path      : /
```

```
$results.Users
```

Output:

```
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    : 40SVDDJJTF4XEEXAMPLE2
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
```

- Per i dettagli sull'API, vedere [GetGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMGroupForUser

Il seguente esempio di codice mostra come utilizzare. Get-IAMGroupForUser

Strumenti per PowerShell

Esempio 1: questo esempio restituisce l'elenco dei gruppi IAM a cui **David** appartiene l'utente IAM.

```
Get-IAMGroupForUser -UserName David
```

Output:

```
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  : ZU2E0WMK6WBZ0EXAMPLE3
GroupName : Developers
Path     : /
```

- Per i dettagli sull'API, vedere [ListGroupForUser](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMGroupList

Il seguente esempio di codice mostra come utilizzare. `Get-IAMGroupList`

Strumenti per PowerShell

Esempio 1: Questo esempio restituisce una raccolta di tutti i gruppi IAM definiti nella versione corrente Account AWS.

```
Get-IAMGroupList
```

Output:

```
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   : ZU2E0WMK6WBZOEXAMPLE2
GroupName : Developers
Path      : /

Arn      : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId   : RHNZZGQJ7QHMAEXAMPLE3
GroupName : Testers
Path      : /
```

- Per i dettagli sull'API, vedere [ListGroups](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMGroupPolicy

Il seguente esempio di codice mostra come utilizzare. `Get-IAMGroupPolicy`

Strumenti per PowerShell

Esempio 1: Questo esempio restituisce i dettagli sulla politica in linea incorporata denominata **PowerUserAccess-Testers** per il gruppo **Testers**. La **PolicyDocument** proprietà è codificata in URL. In questo esempio viene decodificata con il **UrlDecode** metodo.NET.

```
$results = Get-IAMGroupPolicy -GroupName Testers -PolicyName PowerUserAccess-Testers
$results
```

Output:

```

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": "*"
    }
  ]
}
```

- Per i dettagli sull'API, vedere [GetGroupPolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMGroupPolicyList

Il seguente esempio di codice mostra come utilizzare. `Get-IAMGroupPolicyList`

Strumenti per PowerShell

Esempio 1: Questo esempio restituisce un elenco delle politiche in linea incorporate nel gruppo **Testers**. Per ottenere le politiche gestite allegate al gruppo, utilizzare il comando **Get-IAMAttachedGroupPolicyList**.

```
Get-IAMGroupPolicyList -GroupName Testers
```

Output:

```
Deny-Assume-S3-Role-In-Production
PowerUserAccess-Testers
```

- Per i dettagli sull'API, vedere [ListGroupPolicies](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMInstanceProfile

Il seguente esempio di codice mostra come utilizzare. `Get-IAMInstanceProfile`

Strumenti per PowerShell

Esempio 1: Questo esempio restituisce i dettagli del profilo **ec2instancerole** di istanza denominato definito nell' AWS account corrente.

```
Get-IAMInstanceProfile -InstanceProfileName ec2instancerole
```

Output:

```
Arn                : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate         : 2/17/2015 2:49:04 PM
InstanceProfileId  : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path               : /
Roles              : {ec2instancerole}
```

- Per i dettagli sull'API, vedere [GetInstanceProfile](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMInstanceProfileForRole

Il seguente esempio di codice mostra come utilizzare. `Get-IAMInstanceProfileForRole`

Strumenti per PowerShell

Esempio 1: questo esempio restituisce i dettagli del profilo dell'istanza associato al ruolo **ec2instancerole**.

```
Get-IAMInstanceProfileForRole -RoleName ec2instancerole
```

Output:

```
Arn                : arn:aws:iam::123456789012:instance-profile/
ec2instancerole
```

```
CreateDate      : 2/17/2015 2:49:04 PM
InstanceProfileId : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path            : /
Roles           : {ec2instancerole}
```

- Per i dettagli sull'API, vedere [ListInstanceProfilesForRole](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMInstanceProfileList

Il seguente esempio di codice mostra come utilizzare. `Get-IAMInstanceProfileList`

Strumenti per PowerShell

Esempio 1: Questo esempio restituisce una raccolta dei profili di istanza definiti nella versione corrente Account AWS.

```
Get-IAMInstanceProfileList
```

Output:

```
Arn              : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate       : 2/17/2015 2:49:04 PM
InstanceProfileId : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path            : /
Roles           : {ec2instancerole}
```

- Per i dettagli sull'API, vedere [ListInstanceProfiles](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMLoginProfile

Il seguente esempio di codice mostra come utilizzare. `Get-IAMLoginProfile`

Strumenti per PowerShell

Esempio 1: Questo esempio restituisce la data di creazione della password e se è necessaria una reimpostazione della password per l'utente **IAMDavid**.

```
Get-IAMLoginProfile -UserName David
```

Output:

CreateDate	PasswordResetRequired	UserName
-----	-----	-----
12/10/2014 3:39:44 PM	False	David

- Per i dettagli sull'API, vedere [GetLoginProfile](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMMFADevice

Il seguente esempio di codice mostra come utilizzare. `Get-IAMMFADevice`

Strumenti per PowerShell

Esempio 1: questo esempio restituisce i dettagli sul dispositivo MFA assegnato all'utente IAM.

David In questo esempio si può dire che si tratta di un dispositivo virtuale perché **SerialNumber** è un ARN anziché il numero di serie effettivo di un dispositivo fisico.

```
Get-IAMMFADevice -UserName David
```

Output:

EnableDate	SerialNumber	UserName
-----	-----	-----
4/8/2015 9:41:10 AM	arn:aws:iam::123456789012:mfa/David	David

- Per i dettagli sull'API, vedere [ListMfaDevices](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMOpenIDConnectProvider

Il seguente esempio di codice mostra come utilizzare. `Get-IAMOpenIDConnectProvider`

Strumenti per PowerShell

Esempio 1: Questo esempio restituisce i dettagli sul provider OpenID Connect il cui ARN

è. **arn:aws:iam::123456789012:oidc-provider/accounts.google.com** La

ClientIDList proprietà è una raccolta che contiene tutti gli ID client definiti per questo provider.

```
Get-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
arn:aws:iam::123456789012:oidc-provider/oidc.example.com
```

Output:

ClientIDList Url	CreateDate	ThumbprintList
----- ---	-----	-----
{MyOIDCApp} {12345abcdefghijkl67890lmnopqrst98765uvwxyz}	2/3/2015 3:00:30 PM	oidc.example.com

- Per i dettagli sull'API, vedere [GetOpenIdConnectProvider](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMOpenIDConnectProviderList

Il seguente esempio di codice mostra come utilizzare. `Get-IAMOpenIDConnectProviderList`

Strumenti per PowerShell

Esempio 1: Questo esempio restituisce un elenco di ARNS di tutti i provider OpenID Connect definiti nella Account AWS versione corrente.

```
Get-IAMOpenIDConnectProviderList
```

Output:

```
Arn
---
arn:aws:iam::123456789012:oidc-provider/server.example.com
arn:aws:iam::123456789012:oidc-provider/another.provider.com
```

- Per i dettagli sull'API, vedere [ListOpenIdConnectProviders](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMPolicy

Il seguente esempio di codice mostra come utilizzare. `Get-IAMPolicy`

Strumenti per PowerShell

Esempio 1: questo esempio restituisce i dettagli sulla politica gestita il cui ARN è.

arn:aws:iam::123456789012:policy/MySamplePolicy

```
Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [GetPolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMPolicyList

Il seguente esempio di codice mostra come utilizzare. `Get-IAMPolicyList`

Strumenti per PowerShell

Esempio 1: Questo esempio restituisce una raccolta delle prime tre politiche gestite disponibili nell' AWS account corrente. Poiché non **-scope** è specificato, per **all** impostazione predefinita include sia le politiche AWS gestite che quelle gestite dai clienti.

```
Get-IAMPolicyList -MaxItem 3
```

Output:

```
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        : 5ULJS02FYVPYGEXAMPLE3
PolicyName      : AWSMarketplaceFullAccess
UpdateDate      : 2/11/2015 9:21:45 AM
```

Esempio 2: Questo esempio restituisce una raccolta delle prime due politiche gestite dai clienti disponibili nell'account corrente AWS . Viene utilizzato **-Scope local** per limitare l'output alle sole politiche gestite dal cliente.

```
Get-IAMPolicyList -Scope local -MaxItem 2
```

Output:

```
Arn              : arn:aws:iam::123456789012:policy/MyLocalPolicy
AttachmentCount  : 0
CreateDate       : 2/12/2015 9:39:09 AM
DefaultVersionId : v2
Description      :
IsAttachable     : True
```

```

Path           : /
PolicyId       : SQVCBLC4VA0UCEXAMPLE4
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       : X5JPBLJH2Z2S0EXAMPLE5
PolicyName     : policyforec2instancerole
UpdateDate     : 2/18/2015 8:52:31 AM

```

- Per i dettagli sull'API, vedere [ListPolicies](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMPolicyVersion

Il seguente esempio di codice mostra come utilizzare `Get-IAMPolicyVersion`

Strumenti per PowerShell

Esempio 1: questo esempio restituisce il documento relativo alla politica per la **v2** versione della politica il cui ARN è **arn:aws:iam::123456789012:policy/MyManagedPolicy** Il documento di policy contenuto nella **Document** proprietà è codificato in URL e viene decodificato in questo esempio con il **UrlDecode** metodo.NET.

```

$results = Get-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/
MyManagedPolicy -VersionId v2
$results

```

Output:

```

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": "*"
  }
}
```

- Per i dettagli sull'API, vedere [GetPolicyVersion](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMPolicyVersionList

Il seguente esempio di codice mostra come utilizzare `Get-IAMPolicyVersionList`

Strumenti per PowerShell

Esempio 1: questo esempio restituisce l'elenco delle versioni disponibili della politica il cui ARN è **arn:aws:iam::123456789012:policy/MyManagedPolicy**. Per ottenere il documento relativo alla policy per una versione specifica, utilizzate il **Get-IAMPolicyVersion** comando e specificate quella desiderata. **VersionId**

```
Get-IAMPolicyVersionList -PolicyArn arn:aws:iam::123456789012:policy/MyManagedPolicy
```

Output:

CreateDate VersionId	Document	IsDefaultVersion
----- ----- 2/12/2015 9:39:53 AM v2	-----	True
2/12/2015 9:39:09 AM v1		False

- Per i dettagli sull'API, vedere [ListPolicyVersions](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMRole

Il seguente esempio di codice mostra come utilizzare `Get-IAMRole`

Strumenti per PowerShell

Esempio 1: Questo esempio restituisce i dettagli di `lambda_exec_role`. Include il documento sulla politica di fiducia che specifica chi può assumere questo ruolo. Il documento di policy è codificato tramite URL e può essere decodificato utilizzando il metodo `NET.UrlDecode`. In questo esempio, la policy originale aveva rimosso tutti gli spazi bianchi prima di essere caricata nella policy. Per visualizzare i documenti relativi alle politiche in materia di autorizzazioni che determinano cosa può fare qualcuno che assume il ruolo, utilizza i documenti per le politiche in linea e `Get-IAMRolePolicy` `Get-IAMPolicyVersion` per le politiche gestite allegate.

```
$results = Get-IamRole -RoleName lambda_exec_role
$results | Format-List
```

Output:

```
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
```

Output:

```
{"Version":"2012-10-17","Statement":[{"Sid":"","Effect":"Allow","Principal":{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}
```

- Per i dettagli sull'API, vedere [GetRole](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMRoleList

Il seguente esempio di codice mostra come utilizzare. `Get-IAMRoleList`

Strumenti per PowerShell

Esempio 1: Questo esempio recupera un elenco di tutti i ruoli IAM in. Account AWS

```
Get-IAMRoleList
```

Esempio 2: Questo frammento di codice di esempio recupera un elenco di ruoli IAM nell' AWS account e li visualizza tre alla volta, e attende che tu premi Invio tra ogni gruppo. Passa il **Marker** valore della chiamata precedente per specificare dove deve iniziare il gruppo successivo.

```
$nextMarker = $null
Do
{
    $results = Get-IAMRoleList -MaxItem 3 -Marker $nextMarker
    $nextMarker = $AWSHistory.LastServiceResponse.Marker
    $results
    Read-Host
} while ($nextMarker -ne $null)
```

- Per i dettagli sull'API, vedere [ListRoles](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMRolePolicy

Il seguente esempio di codice mostra come utilizzare. `Get-IAMRolePolicy`

Strumenti per PowerShell

Esempio 1: questo esempio restituisce il documento sulla politica delle autorizzazioni per la policy denominata **oneClick_lambda_exec_role_policy** che è incorporata nel ruolo **lambda_exec_role** IAM. Il documento di policy risultante è codificato in URL. In questo esempio viene decodificato con il **UrlDecode** metodo.NET.

```
$results = Get-IAMRolePolicy -RoleName lambda_exec_role -PolicyName
oneClick_lambda_exec_role_policy
$results
```

Output:

PolicyDocument	PolicyName
----- ----- %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)
```

Output:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:*"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:*:*:*"
      ]
    }
  ]
}
```

- Per i dettagli sull'API, vedere [GetRolePolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMRolePolicyList

Il seguente esempio di codice mostra come utilizzare `Get-IAMRolePolicyList`

Strumenti per PowerShell

Esempio 1: questo esempio restituisce l'elenco dei nomi delle politiche in linea incorporate nel ruolo **lambda_exec_role** IAM. Per visualizzare i dettagli di una politica in linea, usa il comando.

Get-IAMRolePolicy

```
Get-IAMRolePolicyList -RoleName lambda_exec_role
```

Output:

```
oneClick_lambda_exec_role_policy
```

- Per i dettagli sull'API, vedere [ListRolePolicies](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMRoleTagList

Il seguente esempio di codice mostra come utilizzare `Get-IAMRoleTagList`

Strumenti per PowerShell

Esempio 1: questo esempio recupera il tag associato al ruolo.

```
Get-IAMRoleTagList -RoleName MyRoleName
```

- Per i dettagli sull'API, vedere [ListRoleTags](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMSAMLProvider

Il seguente esempio di codice mostra come utilizzare `Get-IAMSAMLProvider`

Strumenti per PowerShell

Esempio 1: questo esempio recupera i dettagli sul provider SAML 2.0 il cui ARN è `arn:aws:iam: :123456789012:SAML-provider/SAMLadfs`. La risposta include il documento di metadati che hai ricevuto dal provider di identità per creare l'entità del provider SAML, nonché le date di creazione e scadenza. AWS

```
Get-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/SAMLADFS
```


Output:

```

CreateDate                               SAMLMetadataDocument
ValidUntil
-----
-----
12/23/2014 12:16:55 PM <EntityDescriptor ID="_12345678-1234-5678-9012-example1...
12/23/2114 12:16:54 PM

```

- Per i dettagli sull'API, vedere [GetSamlProvider](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMSAMLProviderList

Il seguente esempio di codice mostra come utilizzare `Get-IAMSAMLProviderList`

Strumenti per PowerShell

Esempio 1: questo esempio recupera l'elenco dei provider SAML 2.0 creati nella versione corrente. Account AWS Restituisce l'ARN, la data di creazione e la data di scadenza per ogni provider SAML.

```
Get-IAMSAMLProviderList
```

Output:

```

Arn                                       CreateDate
ValidUntil
---                                       -
-----
arn:aws:iam::123456789012:saml-provider/SAMLADFS 12/23/2014 12:16:55 PM
12/23/2114 12:16:54 PM

```

- Per i dettagli sull'API, vedere [ListSAMLProviders](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-IAMServerCertificate

Il seguente esempio di codice mostra come utilizzare `Get-IAMServerCertificate`

Strumenti per PowerShell

Esempio 1: Questo esempio recupera i dettagli sul certificato del server denominato **MyServerCertificate**. È possibile trovare i dettagli del certificato nelle proprietà **CertificateBody** e **ServerCertificateMetadata**.

```
$result = Get-IAMServerCertificate -ServerCertificateName MyServerCertificate
$result | format-list
```

Output:

```
CertificateBody          : -----BEGIN CERTIFICATE-----

MIICiTCCAfICCD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMakGA1UEBhMC
VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBA5TC01BTSBDb25zb2x1MRIwEAYDVQQDEw1UZXR0Q21sYWxhZAd
BgkqhkiG9w0BCQEWEG5vb251QGftYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMakGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYD
VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBA5TC01BTSBDb25z
b2x1MRIwEAYDVQQDEw1UZXR0Q21sYWxhZAdBgkqhkiG9w0BCQEWEG5vb251QGft
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMak0dn
+a4GmWIWJ
f0wYK8m9T
21uUSfwfEvySWtC2XADZ4nB+BLygVIk60CpiwsZ3G93vUEI03IyNoH/
rDHudUZg3qX4waLG5M43q7Wgc/
MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb30hjZnzcVQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9q6q
+auNkyExzyLwax1Aoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJI1J00zbhNYS5f6GuoEDmFJ10ZxBHjJnyp3780D8uTs7fLvJx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----

CertificateChain        :
```

```
ServerCertificateMetadata :  
Amazon.IdentityManagement.Model.ServerCertificateMetadata
```

```
$result.ServerCertificateMetadata
```

Output:

```
Arn : arn:aws:iam::123456789012:server-certificate/0rg1/0rg2/  
MyServerCertificate  
Expiration : 1/14/2018 9:52:36 AM  
Path : /0rg1/0rg2/  
ServerCertificateId : ASCAJIFEXAMPLE17HQZYW  
ServerCertificateName : MyServerCertificate  
UploadDate : 4/21/2015 11:14:16 AM
```

- Per i dettagli sull'API, vedere [GetServerCertificate](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMServerCertificateList

Il seguente esempio di codice mostra come utilizzare `Get-IAMServerCertificateList`

Strumenti per PowerShell

Esempio 1: Questo esempio recupera l'elenco dei certificati server che sono stati caricati nella versione corrente Account AWS.

```
Get-IAMServerCertificateList
```

Output:

```
Arn : arn:aws:iam::123456789012:server-certificate/0rg1/0rg2/  
MyServerCertificate  
Expiration : 1/14/2018 9:52:36 AM  
Path : /0rg1/0rg2/  
ServerCertificateId : ASCAJIFEXAMPLE17HQZYW  
ServerCertificateName : MyServerCertificate  
UploadDate : 4/21/2015 11:14:16 AM
```

- Per i dettagli sull'API, vedere [ListServerCertificates](#) in AWS Tools for PowerShell Cmdlet Reference.

Get- IAMServiceLastAccessedDetail

Il seguente esempio di codice mostra come utilizzare. `Get- IAMServiceLastAccessedDetail`

Strumenti per PowerShell

Esempio 1: questo esempio fornisce i dettagli dell'ultimo accesso al servizio da parte dell'entità IAM (utente, gruppo, ruolo o policy) associata alla chiamata Request.

```
Request- IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

Output:

```
f0b7a819-eab0-929b-dc26-ca598911cb9f
```

```
Get- IAMServiceLastAccessedDetail -JobId f0b7a819-eab0-929b-dc26-ca598911cb9f
```

- Per i dettagli sull'API, vedere [GetServiceLastAccessedDetails](#) in AWS Tools for PowerShell Cmdlet Reference.

Get- IAMServiceLastAccessedDetailWithEntity

Il seguente esempio di codice mostra come utilizzare. `Get- IAMServiceLastAccessedDetailWithEntity`

Strumenti per PowerShell

Esempio 1: questo esempio fornisce il timestamp dell'ultimo accesso per il servizio contenuto nella richiesta della rispettiva entità IAM.

```
$results = Get- IAMServiceLastAccessedDetailWithEntity -JobId f0b7a819-eab0-929b-dc26-ca598911cb9f -ServiceNamespace ec2  
$results
```

Output:

```
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
```

Output:

```

EntityInfo                               LastAuthenticated
-----
Amazon.IdentityManagement.Model.EntityInfo 11/16/19 3:47:00 PM

```

```
$results.EntityInfo
```

Output:

```

Arn   : arn:aws:iam::123456789012:user/TestUser
Id    : AIDA4NBK5CXF5TZHU1234
Name  : TestUser
Path  : /
Type  : USER

```

- Per i dettagli sull'API, vedere [GetServiceLastAccessedDetailsWithEntities](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMSigningCertificate

Il seguente esempio di codice mostra come utilizzare `Get-IAMSigningCertificate`

Strumenti per PowerShell

Esempio 1: Questo esempio recupera i dettagli sul certificato di firma associato all'utente denominato **Bob**.

```
Get-IAMSigningCertificate -UserName Bob
```

Output:

```

CertificateBody : -----BEGIN CERTIFICATE-----
                MIICiTCCAFICCCQD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
                VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
                b24xFDASBgNVBAwTC01BTSBDb25zb2x1MRIwEAYDVQQDEw1UZXR0Q21sYWxhZAd
                BgkqhkiG9w0BCQEWEG5vb25lQGFTYXpvbi5jb20wHhcNMTEwNDI0MjA0NTIxWhcN
                MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYD
                VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAwTC01BTSBDb25z
                b2x1MRIwEAYDVQQDEw1UZXR0Q21sYWxhZAdBgkqhkiG9w0BCQEWEG5vb25lQGFT
                YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
                21uUSfwfEvySwTc2XADZ4nB+BLyGVIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9T
                rDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
                Ibb30hjZnzcvcQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
                FFBjvSfpJI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjStB
                NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                -----END CERTIFICATE-----
CertificateId   : Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
Status         : Active
UploadDate    : 4/20/2015 1:26:01 PM
UserName      : Bob

```

- Per i dettagli sull'API, vedere [ListSigningCertificates](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMUser

Il seguente esempio di codice mostra come utilizzare. Get-IAMUser

Strumenti per PowerShell

Esempio 1: Questo esempio recupera i dettagli sull'utente denominato **David**.

```
Get-IAMUser -UserName David
```

Output:

```

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
```

Esempio 2: questo esempio recupera i dettagli sull'utente IAM attualmente connesso.

```
Get-IAMUser
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [GetUser](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-IAMUserList

Il seguente esempio di codice mostra come utilizzare. `Get-IAMUserList`

Strumenti per PowerShell

Esempio 1: Questo esempio recupera una raccolta di utenti nella cartella corrente Account AWS.

```
Get-IAMUserList
```

Output:

```
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
  
```

- Per i dettagli sull'API, vedere [ListUsers](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMUserPolicy

Il seguente esempio di codice mostra come utilizzare `Get-IAMUserPolicy`

Strumenti per PowerShell

Esempio 1: Questo esempio recupera i dettagli della policy in linea denominata **Davids_IAM_Admin_Policy** incorporata nell'utente IAM denominato **David**. Il documento di policy è codificato in URL.

```

$results = Get-IAMUserPolicy -PolicyName Davids_IAM_Admin_Policy -UserName David
$results
  
```

Output:

```

PolicyDocument                                     PolicyName
-----
-----
%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": [  
        "*" ]  
    ]  
  }  
]
```

- Per i dettagli sull'API, vedere [GetUserPolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMUserPolicyList

Il seguente esempio di codice mostra come utilizzare `Get-IAMUserPolicyList`

Strumenti per PowerShell

Esempio 1: Questo esempio recupera l'elenco dei nomi delle politiche in linea incorporate nell'utente IAM denominato **David**

```
Get-IAMUserPolicyList -UserName David
```

Output:

```
 Davids_IAM_Admin_Policy
```

- Per i dettagli sull'API, vedere [ListUserPolicies](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMUserTagList

Il seguente esempio di codice mostra come utilizzare `Get-IAMUserTagList`

Strumenti per PowerShell

Esempio 1: questo esempio recupera il tag associato all'utente.

```
Get-IAMUserTagList -UserName joe
```

- Per i dettagli sull'API, vedere [ListUserTags](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-IAMVirtualMFADevice

Il seguente esempio di codice mostra come utilizzare. `Get-IAMVirtualMFADevice`

Strumenti per PowerShell

Esempio 1: questo esempio recupera una raccolta di dispositivi MFA virtuali assegnati agli utenti dell'account. AWS La **User** proprietà di ciascuno è un oggetto con i dettagli dell'utente IAM a cui è assegnato il dispositivo.

```
Get-IAMVirtualMFADevice -AssignmentStatus Assigned
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [ListVirtualMfaDevices](#) in AWS Tools for PowerShell Cmdlet Reference.

New-IAMAccessKey

Il seguente esempio di codice mostra come utilizzare. `New-IAMAccessKey`

Strumenti per PowerShell

Esempio 1: Questo esempio crea una nuova chiave di accesso e una coppia di chiavi di accesso segrete e le assegna all'utente **David**. Assicurati di salvare i **SecretAccessKey** valori **AccessKeyId** and in un file perché è l'unica volta che puoi ottenere il **SecretAccessKey**. Non puoi recuperarla in un secondo momento. Se si perde la chiave segreta, è necessario creare una nuova coppia di chiavi di accesso.

```
New-IAMAccessKey -UserName David
```

Output:

```
AccessKeyId      : AKIAIOSFODNN7EXAMPLE
CreateDate       : 4/13/2015 1:00:42 PM
SecretAccessKey  : wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY
Status           : Active
UserName         : David
```

- Per i dettagli sull'API, vedere [CreateAccessKey](#) in AWS Tools for PowerShell Cmdlet Reference.

New-IAMAccountAlias

Il seguente esempio di codice mostra come utilizzare. `New-IAMAccountAlias`

Strumenti per PowerShell

Esempio 1: questo esempio modifica l'alias AWS dell'account in **mycompanyaws**. L'indirizzo della pagina di accesso dell'utente cambia in `https://mycompanyaws.signin.aws.amazon.com/console`. L'URL originale che utilizza il numero ID dell'account anziché l'alias (`https://<accountidnumber>.signin.aws.amazon.com/console`) continua a funzionare. Tuttavia, tutti gli URL basati su alias precedentemente definiti smettono di funzionare.

```
New-IAMAccountAlias -AccountAlias mycompanyaws
```

- Per i dettagli sull'API, vedere [CreateAccountAlias](#) in AWS Tools for PowerShell Cmdlet Reference.

New-IAMGroup

Il seguente esempio di codice mostra come utilizzare. `New-IAMGroup`

Strumenti per PowerShell

Esempio 1: Questo esempio crea un nuovo gruppo IAM denominato **Developers**.

```
New-IAMGroup -GroupName Developers
```

Output:

```
Arn      : arn:aws:iam::123456789012:group/Developers
CreateDate : 4/14/2015 11:21:31 AM
GroupId   : QNEJ5PM4NFSQCEXAMPLE1
GroupName : Developers
Path      : /
```

- Per i dettagli sull'API, vedere [CreateGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

New-IAMInstanceProfile

Il seguente esempio di codice mostra come utilizzare `New-IAMInstanceProfile`

Strumenti per PowerShell

Esempio 1: questo esempio crea un nuovo profilo di istanza IAM denominato **ProfileForDevEC2Instance**. È necessario eseguire il **Add-IAMRoleToInstanceProfile** comando separatamente per associare il profilo dell'istanza a un ruolo IAM esistente che fornisce le autorizzazioni all'istanza. Infine, collega il profilo dell'istanza a un'istanza EC2 quando la avvii. A tale scopo, utilizza il **New-EC2Instance** cmdlet con il **InstanceProfile_Arn** parametro or. **InstanceProfile_Name**

```
New-IAMInstanceProfile -InstanceProfileName ProfileForDevEC2Instance
```

Output:

```
Arn      : arn:aws:iam::123456789012:instance-profile/
ProfileForDevEC2Instance
CreateDate : 4/14/2015 11:31:39 AM
InstanceProfileId : DYMFXL556EY46EXAMPLE1
InstanceProfileName : ProfileForDevEC2Instance
Path      : /
Roles     : {}
```

- Per i dettagli sull'API, vedere [CreateInstanceProfile](#) in AWS Tools for PowerShell Cmdlet Reference.

New-IAMLoginProfile

Il seguente esempio di codice mostra come utilizzare `New-IAMLoginProfile`

Strumenti per PowerShell

Esempio 1: Questo esempio crea una password (temporanea) per l'utente IAM di nome Bob e imposta il flag che richiede all'utente di modificare la password al **Bob** successivo accesso.

```
New-IAMLoginProfile -UserName Bob -Password P@ssw0rd -PasswordResetRequired $true
```

Output:

CreateDate	PasswordResetRequired	UserName
-----	-----	-----
4/14/2015 12:26:30 PM	True	Bob

- Per i dettagli sull'API, vedere [CreateLoginProfile](#) in AWS Tools for PowerShell Cmdlet Reference.

New- IAMOpenIDConnectProvider

Il seguente esempio di codice mostra come utilizzare. New- IAMOpenIDConnectProvider

Strumenti per PowerShell

Esempio 1: Questo esempio crea un provider IAM OIDC associato al servizio provider compatibile con OIDC che si trova nell'URL **https://example.oidcprovider.com** e nell'ID client. **my-testapp-1** Il provider OIDC fornisce l'impronta digitale. Per autenticare l'impronta personale, segui la procedura all'indirizzo <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
```

Output:

```
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- Per i dettagli sull'API, vedere [CreateOpenIdConnectProvider](#) in AWS Tools for PowerShell Cmdlet Reference.

New-IAMPolicy

Il seguente esempio di codice mostra come utilizzare. `New-IAMPolicy`

Strumenti per PowerShell

Esempio 1: Questo esempio crea una nuova policy IAM nell' AWS account corrente denominato **MySamplePolicy** Il file **MySamplePolicy.json** fornisce il contenuto della policy. Si noti che è necessario utilizzare il parametro **-Raw** switch per elaborare correttamente il file di policy JSON.

```
New-IAMPolicy -PolicyName MySamplePolicy -PolicyDocument (Get-Content -Raw
MySamplePolicy.json)
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [CreatePolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

New-IAMPolicyVersion

Il seguente esempio di codice mostra come utilizzare. `New-IAMPolicyVersion`

Strumenti per PowerShell

Esempio 1: questo esempio crea una nuova versione «v2" della policy IAM il cui ARN è **arn:aws:iam::123456789012:policy/MyPolicy** e la rende la versione predefinita. Il **NewPolicyVersion.json** file fornisce il contenuto della policy. Si noti che è necessario utilizzare il parametro **-Raw** switch per elaborare correttamente il file di policy JSON.

```
New-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -
PolicyDocument (Get-content -Raw NewPolicyVersion.json) -SetAsDefault $true
```

Output:

CreateDate VersionId	Document	IsDefaultVersion
----- ----- 4/15/2015 10:54:54 AM v2	-----	----- True

- Per i dettagli sull'API, vedere [CreatePolicyVersion](#) in AWS Tools for PowerShell Cmdlet Reference.

New-IAMRole

Il seguente esempio di codice mostra come utilizzare `New-IAMRole`

Strumenti per PowerShell

Esempio 1: Questo esempio crea un nuovo ruolo denominato **MyNewRole** e gli allega la politica trovata nel file **NewRoleTrustPolicy.json**. Si noti che è necessario utilizzare il parametro **-Raw** switch per elaborare correttamente il file di policy JSON. Il documento di policy visualizzato nell'output è codificato in URL. In questo esempio viene decodificato con il **UrlDecode** metodo.NET.

```
$results = New-IAMRole -AssumeRolePolicyDocument (Get-Content -raw
NewRoleTrustPolicy.json) -RoleName MyNewRole
$results
```

Output:

```
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%20%22Effect%22%3A%20%22Allow%22%2C%0D%0A%20%20%20%20%20%20%20%22Principal%22%3A%20%7B%0D%0A%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%20%7D%2C%0D%0A%20%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"
    }
  ]
}

```

- Per i dettagli sull'API, vedere [CreateRole](#) in AWS Tools for PowerShell Cmdlet Reference.

New-IAMSAMLProvider

Il seguente esempio di codice mostra come utilizzare `New-IAMSAMLProvider`

Strumenti per PowerShell

Esempio 1: questo esempio crea una nuova entità provider SAML in IAM. È denominata **MySAMLProvider** ed è descritta dal documento di metadati SAML presente nel file **SAMLMetaData.xml**, che è stato scaricato separatamente dal sito Web del provider di servizi SAML.

```
New-IAMSAMLProvider -Name MySAMLProvider -SAMLMetadataDocument (Get-Content -Raw
SAMLMetaData.xml)
```

Output:


```
arn:aws:iam::123456789012:saml-provider/MySAMLProvider
```

- Per i dettagli sull'API, vedere [CreateSAMLProvider in Cmdlet Reference](#). AWS Tools for PowerShell

New-IAMServiceLinkedRole

Il seguente esempio di codice mostra come utilizzare `New-IAMServiceLinkedRole`

Strumenti per PowerShell

Esempio 1: Questo esempio crea un ruolo collegato ai servizi per il servizio di scalabilità automatica.

```
New-IAMServiceLinkedRole -AWSServiceName autoscaling.amazonaws.com -CustomSuffix  
RoleNameEndsWithThis -Description "My service-linked role to support autoscaling"
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [CreateServiceLinkedRole](#) AWS Tools for PowerShell

New-IAMUser

Il seguente esempio di codice mostra come utilizzare `New-IAMUser`

Strumenti per PowerShell

Esempio 1: questo esempio crea un utente IAM denominato **Bob**. Se Bob deve accedere alla AWS console, devi eseguire separatamente il comando **New-IAMLoginProfile** per creare un profilo di accesso con una password. Se Bob deve eseguire AWS PowerShell comandi CLI multiplatforma o AWS effettuare chiamate API, è necessario eseguire separatamente **New-IAMAccessKey** il comando per creare le chiavi di accesso.

```
New-IAMUser -UserName Bob
```

Output:

```
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

```

- Per i dettagli sull'API, vedere [CreateUser](#) in AWS Tools for PowerShell Cmdlet Reference.

New-IAMVirtualMFADevice

Il seguente esempio di codice mostra come utilizzare `New-IAMVirtualMFADevice`

Strumenti per PowerShell

Esempio 1: questo esempio crea un nuovo dispositivo MFA virtuale. Le righe 2 e 3 estraggono il **Base32StringSeed** valore necessario al programma software MFA virtuale per creare un account (in alternativa al codice QR). Dopo aver configurato il programma con il valore, ottieni due codici di autenticazione sequenziali dal programma. Infine, utilizza l'ultimo comando per collegare il dispositivo MFA virtuale all'utente IAM **Bob** e sincronizzare l'account con i due codici di autenticazione.

```

$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$SR = New-Object System.IO.StreamReader($Device.Base32StringSeed)
$base32stringseed = $SR.ReadToEnd()
$base32stringseed
CZWZMCQNW4DEXAMPLE3V0UGXJFZYSUW7EXAMPLECR4NJFD65GX2SLUDW2EXAMPLE

```

Output:

```

-- 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

```

Esempio 2: questo esempio crea un nuovo dispositivo MFA virtuale. Le righe 2 e 3 estraggono il **QRCodePNG** valore e lo scrivono in un file. Questa immagine può essere scansionata dal programma software MFA virtuale per creare un account (in alternativa all'immissione manuale del valore `StringSeed Base32`). Dopo aver creato l'account nel tuo programma MFA virtuale, ottieni due codici di autenticazione sequenziali e inseriscili negli ultimi comandi per collegare il dispositivo MFA virtuale all'utente **Bob** IAM e sincronizzare l'account.

```
$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
```

Output:

```
-- Pause here to scan PNG with virtual MFA program to register account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

- Per i dettagli sull'API, vedere [CreateVirtualMfaDevice](#) in Cmdlet Reference.AWS Tools for PowerShell

Publish-IAMServerCertificate

Il seguente esempio di codice mostra come utilizzare `Publish-IAMServerCertificate`

Strumenti per PowerShell

Esempio 1: questo esempio carica un nuovo certificato del server sull'account IAM. I file contenenti il corpo del certificato, la chiave privata e (facoltativamente) la catena di certificati devono essere tutti codificati in PEM. Nota che i parametri richiedono il contenuto effettivo dei file anziché i nomi dei file. È necessario utilizzare il parametro **-Raw** switch per elaborare correttamente il contenuto del file.

```
Publish-IAMServerCertificate -ServerCertificateName MyTestCert -CertificateBody
(Get-Content -Raw server.crt) -PrivateKey (Get-Content -Raw server.key)
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [UploadServerCertificate](#) in AWS Tools for PowerShell Cmdlet Reference.

Publish-IAMSigningCertificate

Il seguente esempio di codice mostra come utilizzare `Publish-IAMSigningCertificate`

Strumenti per PowerShell

Esempio 1: questo esempio carica un nuovo certificato di firma X.509 e lo associa all'utente IAM denominato **Bob**. Il file contenente il corpo del certificato è codificato in PEM. Il **CertificateBody** parametro richiede il contenuto effettivo del file di certificato anziché il nome del file. È necessario utilizzare il parametro **-Raw** switch per elaborare correttamente il file.

```
Publish-IAMSigningCertificate -UserName Bob -CertificateBody (Get-Content -Raw
SampleSigningCert.pem)
```

Output:

```
CertificateBody : -----BEGIN CERTIFICATE-----
MIICiTCCAFICCCQD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMAKGA1UEBhMC
VVMxCzAJBgNVBAGTAldBMRAwDgYDZDQDEwDQYDZDQDEwDQYDZDQDEwDQYDZDQDEw
b24xFDASBgNVBAwTC01BTSBDb25zb2x1MRIwEAYDZDQDEwL1UZjXN0Q21sYWMxHzAd
BgkqhkiG9w0BCQEWEG5vb25lQGFTYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI1MjA0NTIxWjCBiDELMAKGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYD
ZDQDEwDQYDZDQDEwDQYDZDQDEwDQYDZDQDEwDQYDZDQDEwDQYDZDQDEwDQYDZDQDEw
b24xFDASBgNVBAwTC01BTSBDb25zb2x1MRIwEAYDZDQDEwL1UZjXN0Q21sYWMxHzAd
BgkqhkiG9w0BCQEWEG5vb25lQGFTYXpvbi5jb20wZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySWtC2XADZ4nB+BLyGVIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb30hjZnzcVQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAAtCu4
nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJI1J00zbhNYS5f6GuoEDmFJ10ZxBHjJnyp3780D8uTs7fLvJx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----

CertificateId   : Y3EK7RMEXAMPLESV33FCEXAMPLEHMJLU
Status         : Active
UploadDate     : 4/20/2015 1:26:01 PM
UserName       : Bob
```

- Per i dettagli sull'API, vedere [UploadSigningCertificate](#) in AWS Tools for PowerShell Cmdlet Reference.

Register-IAMGroupPolicy

Il seguente esempio di codice mostra come utilizzare Register-IAMGroupPolicy

Strumenti per PowerShell

Esempio 1: questo esempio allega la policy gestita dai clienti denominata **TesterPolicy** al gruppo **Testers** IAM. Gli utenti di quel gruppo sono immediatamente interessati dalle autorizzazioni definite nella versione predefinita di tale policy.

```
Register-IAMGroupPolicy -GroupName Testers -PolicyArn  
arn:aws:iam::123456789012:policy/TesterPolicy
```

Esempio 2: questo esempio allega la policy AWS gestita denominata **AdministratorAccess** al gruppo IAM. **Admins** Gli utenti di quel gruppo sono immediatamente interessati dalle autorizzazioni definite nell'ultima versione di tale policy.

```
Register-IAMGroupPolicy -GroupName Admins -PolicyArn arn:aws:iam::aws:policy/  
AdministratorAccess
```

- Per i dettagli sull'API, vedere [AttachGroupPolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Register-IAMRolePolicy

Il seguente esempio di codice mostra come utilizzare Register-IAMRolePolicy

Strumenti per PowerShell

Esempio 1: questo esempio collega la policy AWS gestita denominata **SecurityAudit** al ruolo **CoSecurityAuditors** IAM. Gli utenti che assumono quel ruolo sono immediatamente interessati dalle autorizzazioni definite nell'ultima versione di tale policy.

```
Register-IAMRolePolicy -RoleName CoSecurityAuditors -PolicyArn  
arn:aws:iam::aws:policy/SecurityAudit
```

- Per i dettagli sull'API, vedere [AttachRolePolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Register-IAMUserPolicy

Il seguente esempio di codice mostra come utilizzare. Register-IAMUserPolicy

Strumenti per PowerShell

Esempio 1: questo esempio AWS allega la policy gestita denominata **AmazonCognitoPowerUser** all'utente **Bob** IAM. L'utente è immediatamente interessato dalle autorizzazioni definite nell'ultima versione di tale policy.

```
Register-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::aws:policy/  
AmazonCognitoPowerUser
```

- Per i dettagli sull'API, vedere [AttachUserPolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-IAMAccessKey

Il seguente esempio di codice mostra come utilizzare. Remove-IAMAccessKey

Strumenti per PowerShell

Esempio 1: Questo esempio elimina la coppia di chiavi di AWS accesso con l'ID chiave **AKIAIOSFODNN7EXAMPLE** dall'utente denominato **Bob**.

```
Remove-IAMAccessKey -AccessKeyId AKIAIOSFODNN7EXAMPLE -UserName Bob -Force
```

- Per i dettagli sull'API, vedere [DeleteAccessKey](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-IAMAccountAlias

Il seguente esempio di codice mostra come utilizzare. Remove-IAMAccountAlias

Strumenti per PowerShell

Esempio 1: Questo esempio rimuove l'alias dell'account dal tuo Account AWS. La pagina di accesso utente con l'alias all'indirizzo <https://mycompanyaws.signin.aws.amazon.com/console> non funziona più. Devi invece utilizzare l'URL originale con il tuo numero Account AWS ID su <https://signin.aws.amazon.com/console>. <accountidnumber>

```
Remove-IAMAccountAlias -AccountAlias mycompanyaws
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [DeleteAccountAlias](#) AWS Tools for PowerShell

Remove-IAMAccountPasswordPolicy

Il seguente esempio di codice mostra come utilizzare. `Remove-IAMAccountPasswordPolicy`

Strumenti per PowerShell

Esempio 1: Questo esempio elimina la politica relativa alle password per Account AWS e ripristina tutti i valori ai valori predefiniti originali. Se attualmente non esiste una politica in materia di password, viene visualizzato il seguente messaggio di errore: Impossibile trovare la politica dell'account con il nome PasswordPolicy .

```
Remove-IAMAccountPasswordPolicy
```

- Per i dettagli sull'API, vedere [DeleteAccountPasswordPolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-IAMClientIDFromOpenIDConnectProvider

Il seguente esempio di codice mostra come utilizzare. `Remove-IAMClientIDFromOpenIDConnectProvider`

Strumenti per PowerShell

Esempio 1: questo esempio rimuove l'ID client **My-TestApp-3** dall'elenco degli ID client associati al provider IAM OIDC il cui ARN è. **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
```

- Per i dettagli sull'API, vedere [RemoveClientIDFromOpenIDConnectProvider](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove - IAMGroup

Il seguente esempio di codice mostra come utilizzare. Remove - IAMGroup

Strumenti per PowerShell

Esempio 1: questo esempio elimina il gruppo IAM denominato **MyTestGroup**. Il primo comando rimuove tutti gli utenti IAM che sono membri del gruppo e il secondo comando elimina il gruppo IAM. Entrambi i comandi funzionano senza alcuna richiesta di conferma.

```
(Get-IAMGroup -GroupName MyTestGroup).Users | Remove-IAMUserFromGroup -GroupName  
MyTestGroup -Force  
Remove-IAMGroup -GroupName MyTestGroup -Force
```

- Per i dettagli sull'API, vedere [DeleteGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove - IAMGroupPolicy

Il seguente esempio di codice mostra come utilizzare. Remove - IAMGroupPolicy

Strumenti per PowerShell

Esempio 1: questo esempio rimuove la policy in linea denominata **TesterPolicy** dal gruppo **Testers** IAM. Gli utenti di quel gruppo perdono immediatamente le autorizzazioni definite in quella politica.

```
Remove-IAMGroupPolicy -GroupName Testers -PolicyName TestPolicy
```

- Per i dettagli sull'API, vedere [DeleteGroupPolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove - IAMInstanceProfile

Il seguente esempio di codice mostra come utilizzare. Remove - IAMInstanceProfile

Strumenti per PowerShell

Esempio 1: questo esempio elimina il profilo di istanza EC2 denominato. **MyAppInstanceProfile** Il primo comando rimuove tutti i ruoli dal profilo dell'istanza, quindi il secondo comando elimina il profilo dell'istanza.


```
(Get-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile).Roles | Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyAppInstanceProfile  
Remove-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile
```

- Per i dettagli sull'API, vedere [DeleteInstanceProfile](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-IAMLoginProfile

Il seguente esempio di codice mostra come utilizzare. Remove-IAMLoginProfile

Strumenti per PowerShell

Esempio 1: questo esempio elimina il profilo di accesso dall'utente IAM denominato **Bob**. Ciò impedisce all'utente di accedere alla console. AWS Non impedisce all'utente di eseguire chiamate AWS CLI o API utilizzando chiavi di AWS accesso che potrebbero essere ancora collegate all'account utente. PowerShell

```
Remove-IAMLoginProfile -UserName Bob
```

- Per i dettagli sull'API, vedere [DeleteLoginProfile](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-IAMOpenIDConnectProvider

Il seguente esempio di codice mostra come utilizzare. Remove-IAMOpenIDConnectProvider

Strumenti per PowerShell

Esempio 1: questo esempio elimina il provider IAM OIDC che si connette al provider. **example.oidcprovider.com** Assicurati di aggiornare o eliminare tutti i ruoli che fanno riferimento a questo provider nell'**Principale** elemento della politica di fiducia del ruolo.

```
Remove-IAMOpenIDConnectProvider -OpenIDConnectProviderArn  
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- Per i dettagli sull'API, vedere [DeleteOpenIdConnectProvider](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-IAMPolicy

Il seguente esempio di codice mostra come utilizzare. `Remove-IAMPolicy`

Strumenti per PowerShell

Esempio 1: questo esempio elimina la politica il cui **arn:aws:iam::123456789012:policy/MySamplePolicy** ARN è. Prima di poter eliminare la policy, è necessario eliminare tutte le versioni tranne quella predefinita eseguendo. **Remove-IAMPolicyVersion** È inoltre necessario scollegare la policy da qualsiasi utente, gruppo o ruolo IAM.

```
Remove-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

Esempio 2: Questo esempio elimina una policy eliminando prima tutte le versioni non predefinite della policy, scollegandola da tutte le entità IAM collegate e infine eliminando la policy stessa. La prima riga recupera l'oggetto della policy. La seconda riga recupera tutte le versioni delle politiche che non sono contrassegnate come versione predefinita in una raccolta, quindi elimina ogni politica nella raccolta. La terza riga recupera tutti gli utenti, i gruppi e i ruoli IAM a cui è associata la policy. Le righe da quattro a sei separano la policy da ogni entità collegata. L'ultima riga utilizza questo comando per rimuovere la politica gestita e la versione predefinita rimanente. L'esempio include il parametro **-Force** switch su qualsiasi riga che lo richieda per sopprimere le richieste di conferma.

```
$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
```

- Per i dettagli sull'API, vedere [DeletePolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-IAMPolicyVersion

Il seguente esempio di codice mostra come utilizzare. `Remove-IAMPolicyVersion`

Strumenti per PowerShell

Esempio 1: questo esempio elimina la versione identificata come **v2** dalla policy il cui **arn:aws:iam::123456789012:policy/MySamplePolicy** ARN è.

```
Remove-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy -VersionID v2
```

Esempio 2: Questo esempio elimina una policy eliminando prima tutte le versioni non predefinite della policy e quindi eliminando la policy stessa. La prima riga recupera l'oggetto della policy. La seconda riga recupera tutte le versioni delle politiche che non sono contrassegnate come predefinite in una raccolta e quindi utilizza questo comando per eliminare ogni politica nella raccolta. L'ultima riga rimuove la policy stessa e la versione predefinita rimanente. Tieni presente che per eliminare correttamente una policy gestita, devi anche scollegare la policy da qualsiasi utente, gruppo o ruolo utilizzando i **Unregister-IAMRolePolicy** comandi **Unregister-IAMUserPolicy****Unregister-IAMGroupPolicy**, and. Vedere l'esempio per il **Remove-IAMPolicy** cmdlet.

```
$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
```

- Per i dettagli sull'API, vedere [DeletePolicyVersion](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-IAMRole

Il seguente esempio di codice mostra come utilizzare. `Remove-IAMRole`

Strumenti per PowerShell

Esempio 1: questo esempio elimina il ruolo denominato **MyNewRole** dall'account IAM corrente. Prima di poter eliminare il ruolo, devi prima utilizzare il **Unregister-IAMRolePolicy** comando per scollegare eventuali politiche gestite. Le politiche in linea vengono eliminate con il ruolo.

```
Remove-IAMRole -RoleName MyNewRole
```

Esempio 2: questo esempio rimuove tutte le politiche gestite dal ruolo denominato **MyNewRole** e quindi elimina il ruolo. La prima riga recupera tutte le policy gestite associate al ruolo come raccolta e quindi scollega ogni policy della raccolta dal ruolo. La seconda riga elimina il ruolo stesso. Le politiche in linea vengono eliminate insieme al ruolo.

```
Get-IAMAttachedRolePolicyList -RoleName MyNewRole | Unregister-IAMRolePolicy -  
RoleName MyNewRole  
Remove-IAMRole -RoleName MyNewRole
```

- Per i dettagli sull'API, vedere [DeleteRole](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-IAMRoleFromInstanceProfile

Il seguente esempio di codice mostra come utilizzare `Remove-IAMRoleFromInstanceProfile`

Strumenti per PowerShell

Esempio 1: questo esempio elimina il ruolo denominato **MyNewRole** dal profilo dell'istanza EC2 denominato **MyNewRole**. Un profilo di istanza creato nella console IAM ha sempre lo stesso nome del ruolo, come in questo esempio. Se li crei nell'API o nella CLI, possono avere nomi diversi.

```
Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyNewRole -RoleName MyNewRole  
-Force
```

- Per i dettagli sull'API, vedere [RemoveRoleFromInstanceProfile](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-IAMRolePermissionsBoundary

Il seguente esempio di codice mostra come utilizzare `Remove-IAMRolePermissionsBoundary`

Strumenti per PowerShell

Esempio 1: questo esempio mostra come rimuovere il limite di autorizzazione associato a un ruolo IAM.

```
Remove-IAMRolePermissionsBoundary -RoleName MyRoleName
```

- Per i dettagli sull'API, vedere [DeleteRolePermissionsBoundary](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-IAMRolePolicy

Il seguente esempio di codice mostra come utilizzare. Remove-IAMRolePolicy

Strumenti per PowerShell

Esempio 1: questo esempio elimina la policy in linea **S3AccessPolicy** incorporata nel ruolo IAM. **S3BackupRole**

```
Remove-IAMRolePolicy -PolicyName S3AccessPolicy -RoleName S3BackupRole
```

- Per i dettagli sull'API, vedere [DeleteRolePolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-IAMRoleTag

Il seguente esempio di codice mostra come utilizzare. Remove-IAMRoleTag

Strumenti per PowerShell

Esempio 1: Questo esempio rimuove il tag dal ruolo denominato "MyRoleName" con la chiave del tag «abac». Per rimuovere più tag, fornisci un elenco di chiavi di tag separate da virgole.

```
Remove-IAMRoleTag -RoleName MyRoleName -TagKey "abac","xyzw"
```

- Per i dettagli sull'API, vedere [UntagRole](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-IAMSAMLProvider

Il seguente esempio di codice mostra come utilizzare. Remove-IAMSAMLProvider

Strumenti per PowerShell

Esempio 1: questo esempio elimina il provider IAM SAML 2.0 il cui ARN è.
arn:aws:iam::123456789012:saml-provider/SAMLADFSPROVIDER

```
Remove-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/SAMLADFSPROVIDER
```

- Per i dettagli sull'API, vedere [DeleteSAMLProvider in Cmdlet Reference](#). AWS Tools for PowerShell

Remove-IAMServerCertificate

Il seguente esempio di codice mostra come utilizzare. Remove-IAMServerCertificate
Strumenti per PowerShell

Esempio 1: Questo esempio elimina il certificato del server denominato **MyServerCert**.

```
Remove-IAMServerCertificate -ServerCertificateName MyServerCert
```

- Per i dettagli sull'API, vedere [DeleteServerCertificate](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-IAMServiceLinkedRole

Il seguente esempio di codice mostra come utilizzare. Remove-IAMServiceLinkedRole
Strumenti per PowerShell

Esempio 1: questo esempio ha eliminato il ruolo collegato al servizio. Tieni presente che se il servizio utilizza ancora questo ruolo, questo comando genererà un errore.

```
Remove-IAMServiceLinkedRole -RoleName  
AWSServiceRoleForAutoScaling_RoleNameEndsWithThis
```

- Per i dettagli sull'API, vedere [DeleteServiceLinkedRole](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-IAMSigningCertificate

Il seguente esempio di codice mostra come utilizzare. Remove-IAMSigningCertificate
Strumenti per PowerShell

Esempio 1: questo esempio elimina il certificato di firma con l'ID
Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU dell'utente IAM denominato **Bob**.

```
Remove-IAMSigningCertificate -UserName Bob -CertificateId  
Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
```

- Per i dettagli sull'API, vedere [DeleteSigningCertificate](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-IAMUser

Il seguente esempio di codice mostra come utilizzare `Remove-IAMUser`

Strumenti per PowerShell

Esempio 1: questo esempio elimina l'utente IAM denominato **Bob**.

```
Remove-IAMUser -UserName Bob
```

Esempio 2: questo esempio elimina l'utente IAM denominato **Theresa** insieme a tutti gli elementi che devono essere eliminati per primi.

```
$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
```

- Per i dettagli sull'API, vedere [DeleteUser](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-IAMUserFromGroup

Il seguente esempio di codice mostra come utilizzare `Remove-IAMUserFromGroup`

Strumenti per PowerShell

Esempio 1: questo esempio rimuove l'utente IAM **Bob** dal gruppo **Testers**.

```
Remove-IAMUserFromGroup -GroupName Testers -UserName Bob
```

Esempio 2: questo esempio trova tutti i gruppi di cui l'utente IAM **Theresa** è membro e quindi li rimuove **Theresa** da tali gruppi.

```
$groups = Get-IAMGroupForUser -UserName Theresa
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName -
UserName Theresa -Force }
```

Esempio 3: Questo esempio mostra un modo alternativo per rimuovere l'utente IAM **Bob** dal **Testers** gruppo.

```
Get-IAMGroupForUser -UserName Bob | Remove-IAMUserFromGroup -UserName Bob -GroupName
Testers -Force
```


- Per i dettagli sull'API, vedere [RemoveUserFromGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-IAMUserPermissionsBoundary

Il seguente esempio di codice mostra come utilizzare `Remove-IAMUserPermissionsBoundary`

Strumenti per PowerShell

Esempio 1: questo esempio mostra come rimuovere il limite di autorizzazione associato a un utente IAM.

```
Remove-IAMUserPermissionsBoundary -UserName joe
```

- Per i dettagli sull'API, vedere [DeleteUserPermissionsBoundary](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-IAMUserPolicy

Il seguente esempio di codice mostra come utilizzare `Remove-IAMUserPolicy`

Strumenti per PowerShell

Esempio 1: Questo esempio elimina il nome della policy in linea **AccessToEC2Policy** che è incorporato nell'utente IAM denominato **Bob**

```
Remove-IAMUserPolicy -PolicyName AccessToEC2Policy -UserName Bob
```

Esempio 2: Questo esempio trova tutte le policy in linea incorporate nel nome utente IAM **Theresa** e poi le elimina.

```
$inlinepols = Get-IAMUserPolicies -UserName Theresa  
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName  
Theresa -Force}
```

- Per i dettagli sull'API, vedere [DeleteUserPolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-IAMUserTag

Il seguente esempio di codice mostra come utilizzare `Remove-IAMUserTag`

Strumenti per PowerShell

Esempio 1: Questo esempio rimuove il tag dall'utente chiamato «joe» con la chiave del tag «abac» e «xyzw». Per rimuovere più tag, fornisci un elenco di chiavi di tag separate da virgole.

```
Remove-IAMUserTag -UserName joe -TagKey "abac","xyzw"
```

- Per i dettagli sull'API, vedere [UntagUser](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-IAMVirtualMFADevice

Il seguente esempio di codice mostra come utilizzare `Remove-IAMVirtualMFADevice`

Strumenti per PowerShell

Esempio 1: questo esempio elimina il dispositivo MFA virtuale IAM il cui ARN è.

arn:aws:iam::123456789012:mfa/bob

```
Remove-IAMVirtualMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/bob
```

Esempio 2: questo esempio verifica se all'utente IAM Theresa è assegnato un dispositivo MFA. Se ne viene trovato uno, il dispositivo viene disabilitato per l'utente IAM. Se il dispositivo è virtuale, viene anche eliminato.

```
$mfa = Get-IAMMFADevice -UserName Theresa
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -SerialNumber
$mfa.SerialNumber }
}
```

- Per i dettagli sull'API, vedere [DeleteVirtualMfaDevice](#) in AWS Tools for PowerShell Cmdlet Reference.

Request-IAMCredentialReport

Il seguente esempio di codice mostra come utilizzare `Request-IAMCredentialReport`

Strumenti per PowerShell

Esempio 1: questo esempio richiede la generazione di un nuovo report, operazione che può essere eseguita ogni quattro ore. Se l'ultimo rapporto è ancora recente, viene visualizzato il campo State. **COMPLETE** Utilizzare **Get-IAMCredentialReport** per visualizzare il rapporto completato.

```
Request-IAMCredentialReport
```

Output:

Description	State
-----	----
No report exists. Starting a new report generation task	STARTED

- Per i dettagli sull'API, vedere [GenerateCredentialReport](#) in AWS Tools for PowerShell Cmdlet Reference.

Request-IAMServiceLastAccessedDetail

Il seguente esempio di codice mostra come utilizzare. Request-IAMServiceLastAccessedDetail

Strumenti per PowerShell

Esempio 1: questo esempio è un cmdlet equivalente all'API.

GenerateServiceLastAccessedDetails Ciò fornisce un job id che può essere utilizzato in Get-IAM e ServiceLastAccessedDetail Get-IAM ServiceLastAccessedDetailWithEntity

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

- Per i dettagli sull'API, vedere [GenerateServiceLastAccessedDetails](#) in Cmdlet Reference.AWS Tools for PowerShell

Set-IAMDefaultPolicyVersion

Il seguente esempio di codice mostra come utilizzare. Set-IAMDefaultPolicyVersion

Strumenti per PowerShell

Esempio 1: questo esempio imposta la **v2** versione della policy il cui ARN è **arn:aws:iam::123456789012:policy/MyPolicy** la versione attiva predefinita.

```
Set-IAMDefaultPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -VersionId v2
```

- Per i dettagli sull'API, vedere [SetDefaultPolicyVersion](#) in AWS Tools for PowerShell Cmdlet Reference.

Set-IAMRolePermissionsBoundary

Il seguente esempio di codice mostra come utilizzare. `Set-IAMRolePermissionsBoundary`

Strumenti per PowerShell

Esempio 1: questo esempio mostra come impostare il limite di autorizzazione per un ruolo IAM. È possibile impostare politiche AWS gestite o politiche personalizzate come limite di autorizzazione.

```
Set-IAMRolePermissionsBoundary -RoleName MyRoleName -PermissionsBoundary arn:aws:iam::123456789012:policy/intern-boundary
```

- Per i dettagli sull'API, vedere [PutRolePermissionsBoundary](#) in AWS Tools for PowerShell Cmdlet Reference.

Set-IAMUserPermissionsBoundary

Il seguente esempio di codice mostra come utilizzare. `Set-IAMUserPermissionsBoundary`

Strumenti per PowerShell

Esempio 1: Questo esempio mostra come impostare il limite di autorizzazione per l'utente. È possibile impostare politiche AWS gestite o politiche personalizzate come limite di autorizzazione.

```
Set-IAMUserPermissionsBoundary -UserName joe -PermissionsBoundary arn:aws:iam::123456789012:policy/intern-boundary
```

- Per i dettagli sull'API, vedere [PutUserPermissionsBoundary](#) in AWS Tools for PowerShell Cmdlet Reference.

Sync-IAMMFADevice

Il seguente esempio di codice mostra come utilizzare Sync-IAMMFADevice

Strumenti per PowerShell

Esempio 1: questo esempio sincronizza il dispositivo MFA associato all'**Bob**utente IAM e il cui ARN **arn:aws:iam::123456789012:mfa/bob** è con un programma di autenticazione che ha fornito i due codici di autenticazione.

```
Sync-IAMMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/theresa -  
AuthenticationCode1 123456 -AuthenticationCode2 987654 -UserName Bob
```

Esempio 2: questo esempio sincronizza il dispositivo MFA IAM associato all'**Theresa**utente IAM con un dispositivo fisico che ha il **ABCD12345678** numero di serie e che ha fornito i due codici di autenticazione.

```
Sync-IAMMFADevice -SerialNumber ABCD12345678 -AuthenticationCode1 123456 -  
AuthenticationCode2 987654 -UserName Theresa
```

- Per i dettagli sull'API, vedere [ResyncMfaDevice](#) in AWS Tools for PowerShell Cmdlet Reference.

Unregister-IAMGroupPolicy

Il seguente esempio di codice mostra come utilizzare Unregister-IAMGroupPolicy

Strumenti per PowerShell

Esempio 1: questo esempio separa la politica di gruppo gestita il cui ARN **arn:aws:iam::123456789012:policy/TesterAccessPolicy** proviene dal gruppo denominato **Testers**

```
Unregister-IAMGroupPolicy -GroupName Testers -PolicyArn  
arn:aws:iam::123456789012:policy/TesterAccessPolicy
```

Esempio 2: Questo esempio trova tutte le politiche gestite allegate al gruppo denominato **Testers** e le rimuove dal gruppo.

```
Get-IAMAttachedGroupPolicies -GroupName Testers | Unregister-IAMGroupPolicy -  
Groupname Testers
```

- Per i dettagli sull'API, vedere [DetachGroupPolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Unregister-IAMRolePolicy

Il seguente esempio di codice mostra come utilizzare. `Unregister-IAMRolePolicy`

Strumenti per PowerShell

Esempio 1: questo esempio separa la politica di gruppo gestita il cui ARN **arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy** proviene dal ruolo denominato. **FedTesterRole**

```
Unregister-IAMRolePolicy -RoleName FedTesterRole -PolicyArn  
arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy
```

Esempio 2: questo esempio trova tutte le politiche gestite associate al ruolo denominato **FedTesterRole** e le stacca dal ruolo.

```
Get-IAMAttachedRolePolicyList -RoleName FedTesterRole | Unregister-IAMRolePolicy -  
Rolenam FedTesterRole
```

- Per i dettagli sull'API, vedere [DetachRolePolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Unregister-IAMUserPolicy

Il seguente esempio di codice mostra come utilizzare. `Unregister-IAMUserPolicy`

Strumenti per PowerShell

Esempio 1: questo esempio scollega la policy gestita il cui ARN **arn:aws:iam::123456789012:policy/TesterPolicy** proviene dall'utente IAM denominato. **Bob**

```
Unregister-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::123456789012:policy/  
TesterPolicy
```

Esempio 2: questo esempio trova tutte le policy gestite allegate all'utente IAM denominato **Theresa** e le separa dall'utente.

```
Get-IAMAttachedUserPolicyList -UserName Theresa | Unregister-IAMUserPolicy -Username Theresa
```

- Per i dettagli sull'API, vedere [DetachUserPolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-IAMAccessKey

Il seguente esempio di codice mostra come utilizzare. Update-IAMAccessKey

Strumenti per PowerShell

Esempio 1: Questo esempio modifica lo stato della chiave di accesso **AKIAIOSFODNN7EXAMPLE** per l'utente IAM denominato **Bob** to **Inactive**.

```
Update-IAMAccessKey -UserName Bob -AccessKeyId AKIAIOSFODNN7EXAMPLE -Status Inactive
```

- Per i dettagli sull'API, vedere [UpdateAccessKey](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-IAMAccountPasswordPolicy

Il seguente esempio di codice mostra come utilizzare. Update-IAMAccountPasswordPolicy

Strumenti per PowerShell

Esempio 1: questo esempio aggiorna la politica delle password per l'account con le impostazioni specificate. Si noti che tutti i parametri non inclusi nel comando non vengono lasciati invariati. Vengono invece ripristinati ai valori predefiniti.

```
Update-IAMAccountPasswordPolicy -AllowUsersToChangePasswords $true -HardExpiry $false -MaxPasswordAge 90 -MinimumPasswordLength 8 -PasswordReusePrevention 20 -RequireLowercaseCharacters $true -RequireNumbers $true -RequireSymbols $true -RequireUppercaseCharacters $true
```

- Per i dettagli sull'API, vedere [UpdateAccountPasswordPolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-IAAssumeRolePolicy

Il seguente esempio di codice mostra come utilizzare. Update-IAAssumeRolePolicy

Strumenti per PowerShell

Esempio 1: questo esempio aggiorna il ruolo IAM denominato **ClientRole** con una nuova policy di fiducia, il cui contenuto proviene dal file **ClientRolePolicy.json**. Si noti che è necessario utilizzare il parametro **-Raw** switch per elaborare correttamente il contenuto del file JSON.

```
Update-IAAssumeRolePolicy -RoleName ClientRole -PolicyDocument (Get-Content -raw ClientRolePolicy.json)
```

- Per i dettagli sull'API, vedere [UpdateAssumeRolePolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-IAMGroup

Il seguente esempio di codice mostra come utilizzare. Update-IAMGroup

Strumenti per PowerShell

Esempio 1: questo esempio rinomina il gruppo IAM **Testers** in **AppTesters**.

```
Update-IAMGroup -GroupName Testers -NewGroupName AppTesters
```

Esempio 2: Questo esempio modifica il percorso del gruppo IAM **AppTesters** in **/Org1/Org2/**. Questo modifica l'ARN del gruppo in. **arn:aws:iam::123456789012:group/Org1/Org2/AppTesters**

```
Update-IAMGroup -GroupName AppTesters -NewPath /Org1/Org2/
```

- Per i dettagli sull'API, vedere [UpdateGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-IAMLoginProfile

Il seguente esempio di codice mostra come utilizzare. Update-IAMLoginProfile

Strumenti per PowerShell

Esempio 1: questo esempio imposta una nuova password temporanea per l'utente **Bob** IAM e richiede all'utente di modificare la password al successivo accesso.

```
Update-IAMLoginProfile -UserName Bob -Password "P@ssw0rd1234" -PasswordResetRequired $true
```

- Per i dettagli sull'API, vedere [UpdateLoginProfile](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-IAMOpenIDConnectProviderThumbprint

Il seguente esempio di codice mostra come utilizzare. Update-IAMOpenIDConnectProviderThumbprint

Strumenti per PowerShell

Esempio 1: questo esempio aggiorna l'elenco delle impronte digitali dei certificati per il provider OIDC il cui ARN deve utilizzare una nuova impronta **arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com** personale. Il provider OIDC condivide il nuovo valore quando il certificato associato al provider cambia.

```
Update-IAMOpenIDConnectProviderThumbprint -OpenIDConnectProviderArn arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com -ThumbprintList 7359755EXAMPLEabc3060bce3EXAMPLEec4542a3
```

- Per i dettagli sull'API, vedere [UpdateOpenIdConnectProviderThumbprint](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-IAMRole

Il seguente esempio di codice mostra come utilizzare. Update-IAMRole

Strumenti per PowerShell

Esempio 1: questo esempio aggiorna la descrizione del ruolo e il valore della durata massima della sessione (in secondi) per cui è possibile richiedere la sessione di un ruolo.

```
Update-IAMRole -RoleName MyRoleName -Description "My testing role" -  
MaxSessionDuration 43200
```

- Per i dettagli sull'API, vedere [UpdateRole](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-IAMRoleDescription

Il seguente esempio di codice mostra come utilizzare. Update-IAMRoleDescription

Strumenti per PowerShell

Esempio 1: questo esempio aggiorna la descrizione di un ruolo IAM nel tuo account.

```
Update-IAMRoleDescription -RoleName MyRoleName -Description "My testing role"
```

- Per i dettagli sull'API, vedere [UpdateRoleDescription](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-IAMSAMLProvider

Il seguente esempio di codice mostra come utilizzare. Update-IAMSAMLProvider

Strumenti per PowerShell

Esempio 1: questo esempio aggiorna il provider SAML in IAM il cui ARN

arn:aws:iam::123456789012:saml-provider/SAMLADFS è con un nuovo documento di metadati SAML dal file, **SAMLMetaData.xml**. Tieni presente che devi utilizzare il parametro **-Raw** switch per elaborare correttamente il contenuto del file JSON.

```
Update-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/  
SAMLADFS -SAMLMetadataDocument (Get-Content -Raw SAMLMetaData.xml)
```

- Per i dettagli sull'API, vedere [UpdateSamlProvider](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-IAMServerCertificate

Il seguente esempio di codice mostra come utilizzare. Update-IAMServerCertificate

Strumenti per PowerShell

Esempio 1: Questo esempio rinomina il certificato **MyServerCertificate** denominato **MyRenamedServerCertificate**.

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -  
NewServerCertificateName MyRenamedServerCertificate
```

Esempio 2: questo esempio sposta il certificato denominato nel percorso **MyServerCertificate /Org1/Org2/**. Questo modifica l'ARN della risorsa in. **arn:aws:iam::123456789012:server-certificate/Org1/Org2/MyServerCertificate**

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -NewPath /  
Org1/Org2/
```

- Per i dettagli sull'API, vedere [UpdateServerCertificate](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-IAMSigningCertificate

Il seguente esempio di codice mostra come utilizzare. Update-IAMSigningCertificate

Strumenti per PowerShell

Esempio 1: questo esempio aggiorna il certificato associato all'utente IAM denominato **Bob** e il cui ID di certificato serve **Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU** a contrassegnarlo come inattivo.

```
Update-IAMSigningCertificate -CertificateId Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU -  
UserName Bob -Status Inactive
```

- Per i dettagli sull'API, vedere [UpdateSigningCertificate](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-IAMUser

Il seguente esempio di codice mostra come utilizzare. Update-IAMUser

Strumenti per PowerShell

Esempio 1: questo esempio rinomina l'utente IAM **Bob** in **Robert**.

```
Update-IAMUser -UserName Bob -NewUserName Robert
```

Esempio 2: questo esempio modifica il percorso dell'utente IAM **Bob** in **/Org1/Org2/**, il che modifica effettivamente l'ARN dell'utente in. **arn:aws:iam::123456789012:user/Org1/Org2/bob**

```
Update-IAMUser -UserName Bob -NewPath /Org1/Org2/
```

- Per i dettagli sull'API, vedere [UpdateUser](#) in AWS Tools for PowerShell Cmdlet Reference.

Write-IAMGroupPolicy

Il seguente esempio di codice mostra come utilizzare `Write-IAMGroupPolicy`

Strumenti per PowerShell

Esempio 1: Questo esempio crea una policy in linea denominata **AppTesterPolicy** e la incorpora nel gruppo IAM. **AppTesters** Se esiste già una politica in linea con lo stesso nome, viene sovrascritta. Il contenuto della policy JSON viene fornito nel file. **apptesterpolicy.json** Si noti che è necessario utilizzare il **-Raw** parametro per elaborare correttamente il contenuto del file JSON.

```
Write-IAMGroupPolicy -GroupName AppTesters -PolicyName AppTesterPolicy -  
PolicyDocument (Get-Content -Raw apptesterpolicy.json)
```

- Per i dettagli sull'API, vedere [PutGroupPolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Write-IAMRolePolicy

Il seguente esempio di codice mostra come utilizzare `Write-IAMRolePolicy`

Strumenti per PowerShell

Esempio 1: questo esempio crea una policy in linea denominata **FedTesterRolePolicy** e la incorpora nel ruolo IAM. **FedTesterRole** Se esiste già una politica in linea con lo stesso nome, viene sovrascritta. Il contenuto della policy JSON proviene dal file. **FedTesterPolicy.json** Si

noti che è necessario utilizzare il **-Raw** parametro per elaborare correttamente il contenuto del file JSON.

```
Write-IAMRolePolicy -RoleName FedTesterRole -PolicyName FedTesterRolePolicy -  
PolicyDocument (Get-Content -Raw FedTesterPolicy.json)
```

- Per i dettagli sull'API, vedere [PutRolePolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Write-IAMUserPolicy

Il seguente esempio di codice mostra come utilizzare `Write-IAMUserPolicy`

Strumenti per PowerShell

Esempio 1: questo esempio crea una policy in linea denominata **EC2AccessPolicy** e la incorpora nell'utente IAM. **Bob** Se esiste già una politica in linea con lo stesso nome, viene sovrascritta. Il contenuto della policy JSON proviene dal file. **EC2AccessPolicy.json** Si noti che è necessario utilizzare il **-Raw** parametro per elaborare correttamente il contenuto del file JSON.

```
Write-IAMUserPolicy -UserName Bob -PolicyName EC2AccessPolicy -PolicyDocument (Get-  
Content -Raw EC2AccessPolicy.json)
```

- Per i dettagli sull'API, vedere [PutUserPolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Esempi di Kinesis con Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell con Kinesis.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Get-KINRecord

Il seguente esempio di codice mostra come utilizzare `Get-KINRecord`.

Strumenti per PowerShell

Esempio 1: Questo esempio mostra come restituire ed estrarre dati da una serie di uno o più record. L'iteratore fornito a `Get-kinRecord` determina la posizione iniziale dei record da restituire, che in questo esempio vengono acquisiti in una variabile, `$records`. È quindi possibile accedere a ogni singolo record indicizzando la raccolta `$records`. Supponendo che i dati nel record siano testo con codifica UTF-8, il comando finale mostra come estrarre i dati dall'oggetto e `MemoryStream` restituirli come testo alla console.

```
$records
$records = Get-KINRecord -ShardIterator "AAAAAAAAAAGIc....9VnbiRNaP"
```

Output:

```
MillisBehindLatest NextShardIterator           Records
-----
0                AAAAAAAAAAERNIq...uDn11HuUs {Key1, Key2}
```

```
$records.Records[0]
```

Output:

```
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())
```

Output:

```
test data from string
```

- Per i dettagli sull'API, vedere [GetRecords](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-KINShardIterator

Il seguente esempio di codice mostra come utilizzare. `Get-KINShardIterator`

Strumenti per PowerShell

Esempio 1: restituisce un iteratore di shard per lo shard e la posizione iniziale specificati. I dettagli degli identificatori di shard e dei numeri di sequenza possono essere ottenuti dall'output del cmdlet `Get-KinStream`, facendo riferimento all'insieme Shards dell'oggetto stream restituito. L'iteratore restituito può essere utilizzato con il cmdlet `Get-KinRecord` per estrarre i record di dati nello shard.

```
Get-KINShardIterator -StreamName "mystream" -ShardId "shardId-000000000000" -  
ShardIteratorType AT_SEQUENCE_NUMBER -StartingSequenceNumber "495598645..."
```

Output:

```
AAAAAAAAAAGIc....9VnbiRNaP
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [GetShardIterator](#) AWS Tools for PowerShell

Get-KINStream

Il seguente esempio di codice mostra come utilizzare. `Get-KINStream`

Strumenti per PowerShell

Esempio 1: restituisce i dettagli dello stream specificato.

```
Get-KINStream -StreamName "mystream"
```

Output:

```
HasMoreShards      : False
RetentionPeriodHours : 24
Shards             : {}
StreamARN          : arn:aws:kinesis:us-west-2:123456789012:stream/mystream
StreamName         : mystream
StreamStatus       : ACTIVE
```

- Per i dettagli sull'API, vedere [DescribeStream](#) in AWS Tools for PowerShell Cmdlet Reference.

New-KINStream

Il seguente esempio di codice mostra come utilizzare. `New-KINStream`

Strumenti per PowerShell

Esempio 1: crea un nuovo stream. Per impostazione predefinita, questo cmdlet non restituisce alcun output, pertanto viene aggiunto lo `PassThru` switch - per restituire il valore fornito al `StreamName` parametro - per un uso successivo.

```
$streamName = New-KINStream -StreamName "mystream" -ShardCount 1 -PassThru
```

- Per i dettagli sull'API, vedere [CreateStream](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-KINStream

Il seguente esempio di codice mostra come utilizzare. `Remove-KINStream`

Strumenti per PowerShell

Esempio 1: elimina lo stream specificato. Viene richiesta una conferma prima dell'esecuzione del comando. Per sopprimere la richiesta di conferma, utilizzare l'interruttore `-Force`.

```
Remove-KINStream -StreamName "mystream"
```

- Per i dettagli sull'API, vedere [DeleteStream](#) in AWS Tools for PowerShell Cmdlet Reference.

Write-KINRecord

Il seguente esempio di codice mostra come utilizzare. `Write-KINRecord`

Strumenti per PowerShell

Esempio 1: scrive un record contenente la stringa fornita al parametro -Text.

```
Write-KINRecord -Text "test data from string" -StreamName "mystream" -PartitionKey  
"Key1"
```

Esempio 2: Scrive un record contenente i dati contenuti nel file specificato. Il file viene trattato come una sequenza di byte, quindi se contiene testo, deve essere scritto con la codifica necessaria prima di utilizzarlo con questo cmdlet.

```
Write-KINRecord -FilePath "C:\TestData.txt" -StreamName "mystream" -PartitionKey  
"Key2"
```

- Per i dettagli sull'API, vedere [PutRecord](#) in Cmdlet Reference.AWS Tools for PowerShell

Esempi di Lambda con Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with Lambda.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Add-LMResourceTag

Il seguente esempio di codice mostra come utilizzare `Add-LMResourceTag`.

Strumenti per PowerShell

Esempio 1: aggiunge i tre tag (Washington, Oregon e California) e i relativi valori associati alla funzione specificata identificata dal relativo ARN.

```
Add-LMResourceTag -Resource "arn:aws:lambda:us-west-2:123456789012:function:MyFunction" -Tag @{ "Washington" = "Olympia"; "Oregon" = "Salem"; "California" = "Sacramento" }
```

- Per i dettagli sull'API, vedere [TagResource](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-LMAccountSetting

Il seguente esempio di codice mostra come utilizzare. Get-LMAccountSetting

Strumenti per PowerShell

Esempio 1: questo esempio viene visualizzato per confrontare il limite dell'account e l'utilizzo dell'account

```
Get-LMAccountSetting | Select-Object
@{Name="TotalCodeSizeLimit";Expression={$_.AccountLimit.TotalCodeSize}},
@{Name="TotalCodeSizeUsed";Expression={$_.AccountUsage.TotalCodeSize}}
```

Output:

```
TotalCodeSizeLimit TotalCodeSizeUsed
-----
80530636800          15078795
```

- Per i dettagli sull'API, vedere [GetAccountSettings](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-LMAlias

Il seguente esempio di codice mostra come utilizzare. Get-LMAlias

Strumenti per PowerShell

Esempio 1: questo esempio recupera i pesi di Routing Config per uno specifico alias della funzione Lambda.

```
Get-LMAlias -FunctionName "MylambdaFunction123" -Name "newlabel1" -Select  
RoutingConfig
```

Output:

```
AdditionalVersionWeights  
-----  
{[1, 0.6]}
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [GetAlias](#) AWS Tools for PowerShell

Get-LMFunctionConcurrency

Il seguente esempio di codice mostra come utilizzare. `Get-LMFunctionConcurrency`

Strumenti per PowerShell

Esempio 1: questo esempio ottiene la concorrenza riservata per la funzione Lambda

```
Get-LMFunctionConcurrency -FunctionName "MylambdaFunction123" -Select *
```

Output:

```
ReservedConcurrentExecutions  
-----  
100
```

- Per i dettagli sull'API, vedere [GetFunctionConcurrency](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-LMFunctionConfiguration

Il seguente esempio di codice mostra come utilizzare. `Get-LMFunctionConfiguration`

Strumenti per PowerShell

Esempio 1: questo esempio restituisce la configurazione specifica della versione di una funzione Lambda.

```
Get-LMFunctionConfiguration -FunctionName "MylambdaFunction123" -Qualifier  
"PowershellAlias"
```

Output:

```
CodeSha256           : uW0W0R7z+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
```

- Per i dettagli sull'API, vedere [GetFunctionConfiguration](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-LMFunctionList

Il seguente esempio di codice mostra come utilizzare. Get-LMFunctionList

Strumenti per PowerShell

Esempio 1: questo esempio mostra tutte le funzioni Lambda con dimensioni di codice ordinate

```
Get-LMFunctionList | Sort-Object -Property CodeSize | Select-Object FunctionName,
RunTime, Timeout, CodeSize
```

Output:

FunctionName CodeSize	Runtime	Timeout
test 243	python2.7	3
MylambdaFunction123 659	python3.8	600
myfuncpython1 675	python3.8	303

- Per i dettagli sull'API, vedere [ListFunctions](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-LMPolicy

Il seguente esempio di codice mostra come utilizzare. Get-LMPolicy

Strumenti per PowerShell

Esempio 1: questo esempio mostra la politica delle funzioni della funzione Lambda

```
Get-LMPolicy -FunctionName test -Select Policy
```

Output:

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

- Per i dettagli sull'API, vedere [GetPolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-LMProvisionedConcurrencyConfig

Il seguente esempio di codice mostra come utilizzare. Get-LMProvisionedConcurrencyConfig

Strumenti per PowerShell

Esempio 1: questo esempio ottiene la configurazione simultanea fornita per l'alias specificato della funzione Lambda.

```
C:\>Get-LMProvisionedConcurrencyConfig -FunctionName "MyLambdaFunction123" -
Qualifier "NewAlias1"
```

Output:

```
AllocatedProvisionedConcurrentExecutions : 0
AvailableProvisionedConcurrentExecutions : 0
LastModified                             : 2020-01-15T03:21:26+0000
RequestedProvisionedConcurrentExecutions : 70
Status                                    : IN_PROGRESS
StatusReason                              :
```

- Per i dettagli sull'API, vedere [GetProvisionedConcurrencyConfig](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-LMProvisionedConcurrencyConfigList

Il seguente esempio di codice mostra come utilizzare. Get-LMProvisionedConcurrencyConfigList

Strumenti per PowerShell

Esempio 1: questo esempio recupera l'elenco delle configurazioni di concorrenza assegnate per una funzione Lambda.

```
Get-LMProvisionedConcurrencyConfigList -FunctionName "MyLambdaFunction123"
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [ListProvisionedConcurrencyConfigs](#) AWS Tools for PowerShell

Get-LMResourceTag

Il seguente esempio di codice mostra come utilizzare. `Get-LMResourceTag`

Strumenti per PowerShell

Esempio 1: recupera i tag e i relativi valori attualmente impostati sulla funzione specificata.

```
Get-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction"
```

Output:

```
Key          Value
---          -
California Sacramento
Oregon       Salem
Washington Olympia
```

- Per i dettagli sull'API, vedere [ListTags](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-LMVersionsByFunction

Il seguente esempio di codice mostra come utilizzare. `Get-LMVersionsByFunction`

Strumenti per PowerShell

Esempio 1: questo esempio restituisce l'elenco delle configurazioni specifiche della versione per ogni versione della funzione Lambda.

```
Get-LMVersionsByFunction -FunctionName "MylambdaFunction123"
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [ListVersionsByFunction](#) in AWS Tools for PowerShell Cmdlet Reference.

New-LMAlias

Il seguente esempio di codice mostra come utilizzare. `New-LMAlias`

Strumenti per PowerShell

Esempio 1: questo esempio crea un nuovo alias Lambda per la versione e la configurazione di routing specificate per specificare la percentuale di richieste di chiamata che riceve.

```
New-LMAlias -FunctionName "MyLambdaFunction123" -
RoutingConfig_AdditionalVersionWeight @{Name="1";Value="0.6"} -Description "Alias for
version 4" -FunctionVersion 4 -Name "PowershellAlias"
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [CreateAlias](#) AWS Tools for PowerShell

Publish-LMFunction

Il seguente esempio di codice mostra come utilizzare. `Publish-LMFunction`

Strumenti per PowerShell

Esempio 1: Questo esempio crea una nuova funzione C# (dotnetcore1.0 runtime) denominata in MyFunction AWS Lambda, che fornisce i file binari compilati per la funzione da un file zip sul file system locale (è possibile utilizzare percorsi relativi o assoluti). Le funzioni C# Lambda specificano il gestore per la funzione utilizzando la designazione: `:Namespace.AssemblyName.ClassName::MethodName` È necessario sostituire in modo appropriato le parti relative al nome dell'assembly (senza il suffisso .dll), allo spazio dei nomi, al nome della classe e al nome del metodo nelle specifiche del gestore. La nuova funzione avrà le variabili di ambiente 'envvar1' e 'envvar2' impostate in base ai valori forniti.

```
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" }
```

Output:

```
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      :
```

Esempio 2: questo esempio è simile a quello precedente, tranne per il fatto che i binari delle funzioni vengono prima caricati in un bucket Amazon S3 (che deve trovarsi nella stessa regione della funzione Lambda prevista) e l'oggetto S3 risultante viene quindi referenziato durante la creazione della funzione.

```
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" }
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [CreateFunction](#) AWS Tools for PowerShell

Publish-LMVersion

Il seguente esempio di codice mostra come utilizzare. Publish-LMVersion

Strumenti per PowerShell

Esempio 1: questo esempio crea una versione per l'istantanea esistente di Lambda Function Code

```
Publish-LMVersion -FunctionName "MylambdaFunction123" -Description "Publishing Existing Snapshot of function code as a new version through Powershell"
```

- Per i dettagli sull'API, vedere [PublishVersion](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-LMAlias

Il seguente esempio di codice mostra come utilizzare. Remove-LMAlias

Strumenti per PowerShell

Esempio 1: Questo esempio elimina la funzione Lambda Alias menzionata nel comando.

```
Remove-LMAlias -FunctionName "MylambdaFunction123" -Name "NewAlias"
```

- Per i dettagli sull'API, vedere [DeleteAlias](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-LMFunction

Il seguente esempio di codice mostra come utilizzare. Remove-LMFunction

Strumenti per PowerShell

Esempio 1: questo esempio elimina una versione specifica di una funzione Lambda

```
Remove-LMFunction -FunctionName "MylambdaFunction123" -Qualifier '3'
```

- Per i dettagli sull'API, vedere [DeleteFunction](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-LMFunctionConcurrency

Il seguente esempio di codice mostra come utilizzare. Remove-LMFunctionConcurrency

Strumenti per PowerShell

Esempio 1: questo esempio rimuove la Function Concurrency della Funzione Lambda.

```
Remove-LMFunctionConcurrency -FunctionName "MylambdaFunction123"
```

- Per i dettagli sull'API, vedere [DeleteFunctionConcurrency](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-LMPermission

Il seguente esempio di codice mostra come utilizzare. Remove-LMPermission

Strumenti per PowerShell

Esempio 1: questo esempio rimuove la politica StatementId della funzione per la funzione Lambda specificata.

```
$policy = Get-LMPolicy -FunctionName "MylambdaFunction123" -Select Policy |  
ConvertFrom-Json | Select-Object -ExpandProperty Statement  
Remove-LMPermission -FunctionName "MylambdaFunction123" -StatementId $policy[0].Sid
```

- Per i dettagli sull'API, vedere [RemovePermission](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-LMProvisionedConcurrencyConfig

Il seguente esempio di codice mostra come utilizzare. Remove-LMProvisionedConcurrencyConfig

Strumenti per PowerShell

Esempio 1: Questo esempio rimuove la configurazione Provisioned Concurrency per un alias specifico.

```
Remove-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -Qualifier  
"NewAlias1"
```

- Per i dettagli sull'API, vedere [DeleteProvisionedConcurrencyConfig](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-LMResourceTag

Il seguente esempio di codice mostra come utilizzare. Remove-LMResourceTag

Strumenti per PowerShell

Esempio 1: rimuove i tag forniti da una funzione. Il cmdlet richiederà una conferma prima di procedere, a meno che non venga specificato lo switch -Force. Viene effettuata una sola chiamata al servizio per rimuovere i tag.

```
Remove-LMResourceTag -Resource "arn:aws:lambda:us-west-2:123456789012:function:MyFunction" -TagKey "Washington","Oregon","California"
```

Esempio 2: rimuove i tag forniti da una funzione. Il cmdlet richiederà una conferma prima di procedere, a meno che non venga specificato lo switch -Force. Una volta effettuata la chiamata al servizio per tag fornito.

```
"Washington","Oregon","California" | Remove-LMResourceTag -Resource "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
```

- Per i dettagli sull'API, vedere [UntagResource](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-LMAlias

Il seguente esempio di codice mostra come utilizzare. Update-LMAlias

Strumenti per PowerShell

Esempio 1: questo esempio aggiorna la configurazione di un alias di funzione Lambda esistente. Aggiorna il RoutingConfiguration valore per spostare il 60% (0,6) del traffico alla versione 1

```
Update-LMAlias -FunctionName "MylambdaFunction123" -Description " Alias for version 2" -FunctionVersion 2 -Name "newlabel1" -RoutingConfig_AdditionalVersionWeight @{Name="1";Value="0.6"}
```

- Per i dettagli sull'API, vedere [UpdateAlias](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-LMFunctionCode

Il seguente esempio di codice mostra come utilizzare. Update-LMFunctionCode

Strumenti per PowerShell

Esempio 1: aggiorna la funzione denominata MyFunction " con il nuovo contenuto nel file zip specificato. Per una funzione Lambda C# .NET Core, il file zip deve contenere l'assembly compilato.

```
Update-LMFunctionCode -FunctionName MyFunction -ZipFilename .\UpdatedCode.zip
```

Esempio 2: questo esempio è simile a quello precedente ma utilizza un oggetto Amazon S3 contenente il codice aggiornato per aggiornare la funzione.

```
Update-LMFunctionCode -FunctionName MyFunction -BucketName mybucket -Key  
UpdatedCode.zip
```

- Per i dettagli sull'API, vedere [UpdateFunctionCode](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-LMFunctionConfiguration

Il seguente esempio di codice mostra come utilizzare. Update-LMFunctionConfiguration

Strumenti per PowerShell

Esempio 1: questo esempio aggiorna la configurazione della funzione Lambda esistente

```
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
```

- Per i dettagli sull'API, vedere [UpdateFunctionConfiguration](#) in AWS Tools for PowerShell Cmdlet Reference.

Write-LMFunctionConcurrency

Il seguente esempio di codice mostra come utilizzare. Write-LMFunctionConcurrency

Strumenti per PowerShell

Esempio 1: Questo esempio applica le impostazioni di concorrenza per la funzione nel suo insieme.

```
Write-LMFunctionConcurrency -FunctionName "MyLambdaFunction123" -
ReservedConcurrentExecution 100
```

- Per i dettagli sull'API, vedere [PutFunctionConcurrency](#) in AWS Tools for PowerShell Cmdlet Reference.

Write-LMProvisionedConcurrencyConfig

Il seguente esempio di codice mostra come utilizzare. Write-LMProvisionedConcurrencyConfig

Strumenti per PowerShell

Esempio 1: Questo esempio aggiunge una configurazione di concorrenza fornita all'alias di una funzione

```
Write-LMProvisionedConcurrencyConfig -FunctionName "MyLambdaFunction123" -
ProvisionedConcurrentExecution 20 -Qualifier "NewAlias1"
```

- Per i dettagli sull'API, vedere [PutProvisionedConcurrencyConfig](#) in AWS Tools for PowerShell Cmdlet Reference.

Esempi di Amazon ML con Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon ML. AWS Tools for PowerShell

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Get-MLBatchPrediction

Il seguente esempio di codice mostra come utilizzare `Get-MLBatchPrediction`.

Strumenti per PowerShell

Esempio 1: restituisce i metadati dettagliati per una previsione in batch con ID ID.

```
Get-MLBatchPrediction -BatchPredictionId ID
```

- Per i dettagli sull'API, vedere [GetBatchPrediction](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-MLBatchPredictionList

Il seguente esempio di codice mostra come utilizzare `Get-MLBatchPredictionList`

Strumenti per PowerShell

Esempio 1: restituisce un elenco di tutti BatchPredictions i record di dati associati che corrispondono al criterio di ricerca fornito nella richiesta.

```
Get-MLBatchPredictionList
```

Esempio 2: restituisce un elenco di tutti BatchPredictions con lo stato COMPLETATO.

```
Get-MLBatchPredictionList -FilterVariable Status -EQ COMPLETED
```

- Per i dettagli sull'API, vedere [DescribeBatchPredictions](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-MLDataSource

Il seguente esempio di codice mostra come utilizzare. Get-MLDataSource

Strumenti per PowerShell

Esempio 1: restituisce i metadati, lo stato e le informazioni sul file di dati per a DataSource con l'ID id

```
Get-MLDataSource -DataSourceId ID
```

- Per i dettagli sull'API, vedere [GetDataSource](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-MLDataSourceList

Il seguente esempio di codice mostra come utilizzare. Get-MLDataSourceList

Strumenti per PowerShell

Esempio 1: restituisce un elenco di tutti i record di dati DataSources e dei relativi record associati.

```
Get-MLDataSourceList
```

Esempio 2: restituisce un elenco di tutti DataSources con lo stato COMPLETATO.

```
Get-MLDataDourceList -FilterVariable Status -EQ COMPLETED
```

- Per i dettagli sull'API, vedere [DescribeDataSources](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-MLEvaluation

Il seguente esempio di codice mostra come utilizzare. Get-MLEvaluation

Strumenti per PowerShell

Esempio 1: restituisce i metadati e lo stato di una valutazione con ID ID.

```
Get-MLEvaluation -EvaluationId ID
```

- Per i dettagli sull'API, vedere [GetEvaluation](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-MLEvaluationList

Il seguente esempio di codice mostra come utilizzare. Get-MLEvaluationList

Strumenti per PowerShell

Esempio 1: restituisce un elenco di tutte le risorse di valutazione

```
Get-MLEvaluationList
```

Esempio 2: restituisce un elenco di tutte le valutazioni con lo stato COMPLETATO.

```
Get-MLEvaluationList -FilterVariable Status -EQ COMPLETED
```

- Per i dettagli sull'API, vedere [DescribeEvaluations](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-MLModel

Il seguente esempio di codice mostra come utilizzare. Get-MLModel

Strumenti per PowerShell

Esempio 1: restituisce i metadati dettagliati, lo stato, lo schema e le informazioni sul file di dati per un MLModel con ID ID.

```
Get-MLModel -ModelId ID
```

- Per i dettagli sull'API, vedere [AWS Tools for PowerShell GetMLModel](#) in Cmdlet Reference.

Get-MLModelList

Il seguente esempio di codice mostra come utilizzare. Get-MLModelList

Strumenti per PowerShell

Esempio 1: restituisce un elenco di tutti i modelli e i relativi record di dati associati.

```
Get-MLModelList
```

Esempio 2: restituisce un elenco di tutti i modelli con lo stato COMPLETATO.

```
Get-MLModelList -FilterVariable Status -EQ COMPLETED
```

- Per i dettagli sull'API, vedere [DescribeMLModels in Cmdlet Reference AWS Tools for PowerShell](#) .

Get-MLPrediction

Il seguente esempio di codice mostra come utilizzare. `Get-MLPrediction`

Strumenti per PowerShell

Esempio 1: invia un record all'URL dell'endpoint di previsione in tempo reale per Model con ID ID.

```
Get-MLPrediction -ModelId ID -PredictEndpoint URL -Record @{"A" = "B"; "C" = "D";}
```

- Per i dettagli sull'API, vedere [Predict](#) in AWS Tools for PowerShell Cmdlet Reference.

New-MLBatchPrediction

Il seguente esempio di codice mostra come utilizzare. `New-MLBatchPrediction`

Strumenti per PowerShell

Esempio 1: crea una nuova richiesta di previsione in batch per il modello con ID ID e inserisci l'output nella posizione S3 specificata.

```
New-MLBatchPrediction -ModelId ID -Name NAME -OutputURI s3://...
```

- Per i dettagli sull'API, vedere [CreateBatchPrediction](#) in AWS Tools for PowerShell Cmdlet Reference.

New-MLDataSourceFromS3

Il seguente esempio di codice mostra come utilizzare. `New-MLDataSourceFromS3`

Strumenti per PowerShell

Esempio 1: crea un'origine dati con dati per una posizione S3, con un nome NAME e uno schema di SCHEMA.

```
New-MLDataSourceFromS3 -Name NAME -ComputeStatistics $true -DataSpec_DataLocationS3 "s3://BUCKET/KEY" -DataSchema SCHEMA
```

- Per i dettagli sull'API, consulta [CreateDataSourceFromS3](#) in AWS Tools for PowerShell Cmdlet Reference.

New-MLEvaluation

Il seguente esempio di codice mostra come utilizzare. `New-MLEvaluation`

Strumenti per PowerShell

Esempio 1: creare una valutazione per un determinato ID di origine dati e un id del modello

```
New-MLEvaluation -Name NAME -DataSourceId DSID -ModelId MID
```

- Per i dettagli sull'API, vedere [CreateEvaluation](#) in AWS Tools for PowerShell Cmdlet Reference.

New-MLModel

Il seguente esempio di codice mostra come utilizzare. `New-MLModel`

Strumenti per PowerShell

Esempio 1: creare un nuovo modello con dati di addestramento.

```
New-MLModel -Name NAME -ModelType BINARY -Parameter @{...} -TrainingDataSourceId ID
```

- Per i dettagli sull'API, vedere [CreateMLModel in Cmdlet Reference](#) AWS Tools for PowerShell .

New-MLRealtimeEndpoint

Il seguente esempio di codice mostra come utilizzare. `New-MLRealtimeEndpoint`

Strumenti per PowerShell

Esempio 1: crea un nuovo endpoint di previsione in tempo reale per l'ID del modello specificato.

```
New-MLRealtimeEndpoint -ModelId ID
```

- Per i dettagli sull'API, vedere [CreateRealtimeEndpoint](#) in AWS Tools for PowerShell Cmdlet Reference.

Esempi di Macie che utilizzano Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with Macie.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Get-MAC2FindingList

Il seguente esempio di codice mostra come utilizzare `Get-MAC2FindingList`.

Strumenti per PowerShell

Esempio 1: restituisce un elenco di risultati contenenti un rilevamento `FindingIds` di dati sensibili di tipo «`CREDIT_CARD_NUMBER`» o «`US_SOCIAL_SECURITY_NUMBER`»

```
$criterionAddProperties = New-Object
    Amazon.Macie2.Model.CriterionAdditionalProperties

$criterionAddProperties.Eq = @(
    "CREDIT_CARD_NUMBER"
    "US_SOCIAL_SECURITY_NUMBER"
)
```

```
$FindingCriterion = @{
  'classificationDetails.result.sensitiveData.detections.type' =
  [Amazon.Macie2.Model.CriterionAdditionalProperties]$criterionAddProperties
}

Get-MAC2FindingList -FindingCriteria_Criterion $FindingCriterion -MaxResult 5
```

- AWS Tools for PowerShell Per [ListFindings](#) dettagli sull'API, vedere in Cmdlet Reference.

AWS OpsWorks esempi di utilizzo di Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with AWS OpsWorks.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

New-OPSDeployment

Il seguente esempio di codice mostra come utilizzare `New-OPSDeployment`.

Strumenti per PowerShell

Esempio 1: questo comando crea una nuova distribuzione di app su tutte le istanze basate su Linux in un livello in Stacks. AWS OpsWorks Anche se specificate un ID di livello, dovete specificare anche un ID stack. Il comando consente alla distribuzione di riavviare le istanze, se necessario.

```
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"}
```

Esempio 2: questo comando distribuisce la **appsetup** ricetta dal **phpapp** ricettario e la **secbaseline** ricetta dal ricettario. **testcookbook** L'obiettivo di implementazione è un'istanza, ma sono necessari anche l'ID dello stack e l'ID del livello. L'**allow_reboot** attributo del parametro **Command_Arg** è impostato su, il che consente alla distribuzione di riavviare le istanze **true**, se necessario.

```
$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"
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [CreateDeployment](#) AWS Tools for PowerShell

Listino prezzi AWS esempi di utilizzo di Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with Listino prezzi AWS.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Get-PLSAttributeValue

Il seguente esempio di codice mostra come utilizzare `Get-PLSAttributeValue`.

Strumenti per PowerShell

Esempio 1: restituisce i valori per l'attributo 'VolumeType' per Amazon EC2 nella regione us-east-1.

```
Get-PLSAttributeValue -ServiceCode AmazonEC2 -AttributeName "volumeType" -region us-east-1
```

Output:

```
Value
-----
Cold HDD
General Purpose
Magnetic
Provisioned IOPS
Throughput Optimized HDD
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [GetAttributeValues](#) AWS Tools for PowerShell

Get-PLSProduct

Il seguente esempio di codice mostra come utilizzare `Get-PLSProduct`

Strumenti per PowerShell

Esempio 1: dettagli sulla restituzione di tutti i prodotti per Amazon EC2.

```
Get-PLSProduct -ServiceCode AmazonEC2 -Region us-east-1
```

Output:

```
{"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"}, ...
```

Esempio 2: restituisce i dati per Amazon EC2 nella regione us-east-1 filtrati per tipi di volume «General Purpose» con supporto SSD.

```
Get-PLSProduct -ServiceCode AmazonEC2 -Filter
@{Type="TERM_MATCH";Field="volumeType";Value="General
Purpose"},@{Type="TERM_MATCH";Field="storageMedia";Value="SSD-backed"} -Region us-
east-1
```

Output:

```
{"product":{"productFamily":"Storage","attributes":{"storageMedia":"SSD-
backed","maxThroughputvolume":"160 MB/sec","volumeType":"General
Purpose","maxIopsvolume":"10000"},...
```

- [GetProducts](#) Per i AWS Tools for PowerShell dettagli sull'API, consulta Cmdlet Reference.

Get-PLSService

Il seguente esempio di codice mostra come utilizzare. Get-PLSService

Strumenti per PowerShell

Esempio 1: restituisce i metadati per tutti i codici di servizio disponibili nella regione us-east-1.

```
Get-PLSService -Region us-east-1
```

Output:

AttributeNames	ServiceCode
----------------	-------------


```

-----
{productFamily, servicecode, groupDescription, termType...}
{productFamily, servicecode, termType, usagetype...}
{productFamily, servicecode, termType, usagetype...}
{productFamily, servicecode, termType, usagetype...}
{productFamily, servicecode, termType, usagetype...}
{productFamily, servicecode, termType, usagetype...}
...
-----
AWSBudgets
AWSCloudTrail
AWSCodeCommit
AWSCodeDeploy
AWSCodePipeline
AWSConfig

```

Esempio 2: restituisce i metadati per il servizio Amazon EC2 nella regione us-east-1.

```
Get-PLSService -ServiceCode AmazonEC2 -Region us-east-1
```

Output:

```

AttributeNames                                     ServiceCode
-----
{volumeType, maxIopsvolume, instanceCapacity10xlarge, locationType...} AmazonEC2

```

- Per i dettagli sull'API, vedere [DescribeServices](#) in Cmdlet Reference.AWS Tools for PowerShell

Esempi di Resource Groups che utilizzano Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando il AWS Tools for PowerShell con Resource Groups.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Add-RGResourceTag

Il seguente esempio di codice mostra come utilizzare `Add-RGResourceTag`.

Strumenti per PowerShell

Esempio 1: Questo esempio aggiunge la chiave di tag «Instances» con valore «workboxes» al gruppo di risorse specificato `arn`

```
Add-RGResourceTag -Tag @{Instances="workboxes"} -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes
```

Output:

```
Arn                                     Tags
---                                     ----
arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes {[Instances,
workboxes]}
```

- Per i dettagli sull'API, vedere [Tag](#) in AWS Tools for PowerShell Cmdlet Reference.

Find-RGResource

Il seguente esempio di codice mostra come utilizzare `Find-RGResource`

Strumenti per PowerShell

Esempio 1: questo esempio crea un `ResourceQuery` tipo di risorsa ad esempio con filtri di tag e trova risorse.

```
$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
```

Output:

```
ResourceArn                                     ResourceType
-----
arn:aws:ec2:eu-west-1:123456789012:instance/i-0123445b6cb7bd67b AWS::EC2::Instance
```

- Per i dettagli sull'API, vedere [SearchResources](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-RGGroup

Il seguente esempio di codice mostra come utilizzare. Get-RGGroup

Strumenti per PowerShell

Esempio 1: Questo esempio recupera il gruppo di risorse in base al nome del gruppo

```
Get-RGGroup -GroupName auto-no
```

Output:

```
Description GroupArn                                     Name
-----
arn:aws:resource-groups:eu-west-1:123456789012:group/auto-no auto-no
```

- Per i dettagli sull'API, vedere [GetGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-RGGroupList

Il seguente esempio di codice mostra come utilizzare. Get-RGGroupList

Strumenti per PowerShell

Esempio 1: questo esempio elenca i gruppi di risorse già creati.

```
Get-RGGroupList
```

Output:

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

- Per i dettagli sull'API, vedere [ListGroups](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-RGGroupQuery

Il seguente esempio di codice mostra come utilizzare. Get-RGGroupQuery

Strumenti per PowerShell

Esempio 1: questo esempio recupera la query di risorse per il gruppo di risorse specificato

```
Get-RGGroupQuery -GroupName auto-no | Select-Object -ExpandProperty ResourceQuery
```

Output:

```
Query
-----
Type
----
{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":[{"Key":"auto","Values":["no"]}]} TAG_FILTERS_1_0
```

- Per i dettagli sull'API, vedere [GetGroupQuery](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-RGGroupResourceList

Il seguente esempio di codice mostra come utilizzare. Get-RGGroupResourceList

Strumenti per PowerShell

Esempio 1: questo esempio elenca le risorse del gruppo in base al tipo di risorsa filtrate

```
Get-RGGroupResourceList -Filter @{Name="resource-type";Values="AWS::EC2::Instance"}
-GroupName auto-yes | Select-Object -ExpandProperty ResourceIdentifiers
```

Output:

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

- Per i dettagli sull'API, vedere [ListGroupResources](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-RGResourceTag

Il seguente esempio di codice mostra come utilizzare. Get-RGResourceTag

Strumenti per PowerShell

Esempio 1: Questo esempio elenca i tag per il gruppo di risorse specificato arn

```
Get-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes
```

Output:

Key	Value
---	-----
Instances	workboxes

- Per i dettagli sull'API, vedere [GetTags](#) in AWS Tools for PowerShell Cmdlet Reference.

New-RGGroup

Il seguente esempio di codice mostra come utilizzare. New-RGGroup

Strumenti per PowerShell

Esempio 1: Questo esempio crea un nuovo gruppo di AWS risorse Resource Groups basato su tag denominato TestPowerShellGroup. Il gruppo include istanze Amazon EC2 nella regione

corrente contrassegnate con la chiave di tag «Name» e il valore del tag «test2». Il comando restituisce la query e il tipo di gruppo e i risultati dell'operazione.

```
$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."
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [CreateGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-RGGroup

Il seguente esempio di codice mostra come utilizzare. Remove-RGGroup

Strumenti per PowerShell

Esempio 1: Questo esempio rimuove il gruppo di risorse denominato

```
Remove-RGGroup -GroupName non-tag-cfn-elbv2
```

Output:

```

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

```

- Per i dettagli sull'API, vedere [DeleteGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-RGResourceTag

Il seguente esempio di codice mostra come utilizzare `Remove-RGResourceTag`

Strumenti per PowerShell

Esempio 1: Questo esempio rimuove il tag menzionato dal gruppo di risorse

```
Remove-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/
workboxes -Key Instances
```

Output:

```

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}

```

- Per i dettagli sull'API, vedere [Untag](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-RGGroup

Il seguente esempio di codice mostra come utilizzare. Update-RGGroup

Strumenti per PowerShell

Esempio 1: questo esempio aggiorna la descrizione del gruppo

```
Update-RGGroup -GroupName auto-yes -Description "Instances auto-remove"
```

Output:

Description	GroupArn
Name	
-----	-----
Instances to be cleaned	arn:aws:resource-groups:eu-west-1:123456789012:group/auto-yes
auto-yes	auto-yes

- Per i dettagli sull'API, vedere [UpdateGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-RGGroupQuery

Il seguente esempio di codice mostra come utilizzare. Update-RGGroupQuery

Strumenti per PowerShell

Esempio 1: Questo esempio crea un oggetto di interrogazione e aggiorna la query per il gruppo.

```
$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
```

Output:


```
GroupName ResourceQuery
-----
build600 Amazon.ResourceGroups.Model.ResourceQuery
```

- Per i dettagli sull'API, vedere [UpdateGroupQuery](#) in AWS Tools for PowerShell Cmdlet Reference.

Esempi di API di tagging dei Resource Groups utilizzando Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando l'API AWS Tools for PowerShell with Resource Groups Tagging.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Add-RGTResourceTag

Il seguente esempio di codice mostra come utilizzare `Add-RGTResourceTag`.

Strumenti per PowerShell

Esempio 1: questo esempio aggiunge le chiavi tag «stage» e «version» con i valori «beta» e «preprod_test» a un bucket Amazon S3 e una tabella Amazon DynamoDB. Viene effettuata una sola chiamata al servizio per applicare i tag.

```
$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" }
```

Esempio 2: questo esempio aggiunge i tag e i valori specificati a un bucket Amazon S3 e a una tabella Amazon DynamoDB. Vengono effettuate due chiamate al servizio, una per ogni risorsa ARN reindirizzata al cmdlet.

```
$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" }
```

- Per i dettagli sull'API, vedere [TagResources](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-RGTResource

Il seguente esempio di codice mostra come utilizzare. Get-RGTResource

Strumenti per PowerShell

Esempio 1: restituisce tutte le risorse etichettate in una regione e le chiavi dei tag associate alla risorsa. Se non viene fornito alcun parametro -Region al cmdlet, questo tenterà di dedurre la regione dalla shell o dai metadati dell'istanza EC2.

```
Get-RGTResource
```

Output:

ResourceARN	Tags
-----	----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable	{stage, version}
arn:aws:s3:::mybucket	{stage, version,
othertag}	

Esempio 2: restituisce tutte le risorse con tag del tipo specificato in una regione. La stringa per ogni nome di servizio e tipo di risorsa è la stessa incorporata nell'Amazon Resource Name (ARN) di una risorsa.

```
Get-RGTResource -ResourceType "s3"
```

Output:

ResourceARN	Tags
-----	----
arn:aws:s3:::mybucket	{stage, version,
othertag}	

Esempio 3: restituisce tutte le risorse etichettate del tipo specificato in una regione. Si noti che quando i tipi di risorse vengono inseriti nel cmdlet, viene effettuata una chiamata al servizio per ogni tipo di risorsa fornito.

```
"dynamodb","s3" | Get-RGTResource
```

Output:

ResourceARN	Tags
-----	----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable	{stage, version}
arn:aws:s3:::mybucket	{stage, version,
othertag}	

Esempio 4: restituisce tutte le risorse etichettate che corrispondono al filtro specificato.

```
Get-RGTResource -TagFilter @{ Key="stage" }
```

Output:

ResourceARN	Tags
-----	----
arn:aws:s3:::mybucket	{stage, version,
othertag}	

Esempio 5: restituisce tutte le risorse con tag che corrispondono al filtro e al tipo di risorsa specificati.

```
Get-RGTResource -TagFilter @{ Key="stage" } -ResourceType "dynamodb"
```

Output:

ResourceARN	Tags
-----	----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable	{stage, version}

Esempio 6: restituisce tutte le risorse con tag che corrispondono al filtro specificato.

```
Get-RGTResource -TagFilter @{ Key="stage"; Values=@("beta","gamma") }
```

Output:

ResourceARN	Tags
-----	----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable	{stage, version}

- Per i dettagli sull'API, vedere [GetResources](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-RGTTagKey

Il seguente esempio di codice mostra come utilizzare. Get-RGTTagKey

Strumenti per PowerShell

Esempio 1: restituisce tutte le chiavi dei tag nella regione specificata. Se il parametro -Region non è specificato, il cmdlet tenterà di dedurre la regione dalla regione della shell predefinita o dai metadati dell'istanza EC2. Tieni presente che le chiavi dei tag non vengono restituite in alcun ordine specifico.

```
Get-RGTTagKey -region us-west-2
```

Output:

```
version
stage
```

- Per i dettagli sull'API, vedere [GetTagKeys](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-RGTTagValue

Il seguente esempio di codice mostra come utilizzare. `Get-RGTTagValue`

Strumenti per PowerShell

Esempio 1: restituisce il valore del tag specificato in una regione. Se il parametro `-Region` non è specificato, il cmdlet tenterà di dedurre la regione dalla regione della shell predefinita o dai metadati dell'istanza EC2.

```
Get-RGTTagValue -Key "stage" -Region us-west-2
```

Output:

```
beta
```

- Per i dettagli sull'API, consulta Cmdlet Reference. [GetTagValues](#) AWS Tools for PowerShell

Remove-RGTResourceTag

Il seguente esempio di codice mostra come utilizzare. `Remove-RGTResourceTag`

Strumenti per PowerShell

Esempio 1: rimuove le chiavi dei tag «stage» e «version» e i valori associati da un bucket Amazon S3 e da una tabella Amazon DynamoDB. Viene effettuata una singola chiamata al servizio per rimuovere i tag. Prima di rimuovere i tag, il cmdlet richiederà una conferma. Per ignorare la conferma, aggiungere il parametro `-Force`.

```
$arn1 = "arn:aws:s3:::mybucket"  
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"  
  
Remove-RGTResourceTag -ResourceARNList $arn1,$arn2 -TagKey "stage","version"
```

Esempio 2: rimuove le chiavi dei tag «stage» e «version» e i valori associati da un bucket Amazon S3 e da una tabella Amazon DynamoDB. Vengono effettuate due chiamate al servizio, una per ogni risorsa ARN reindirizzata al cmdlet. Prima di ogni chiamata, il cmdlet richiederà una conferma. Per ignorare la conferma, aggiungere il parametro `-Force`.

```
$arn1 = "arn:aws:s3:::mybucket"
```

```
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"  
  
$arn1,$arn2 | Remove-RGTResourceTag -TagKey "stage","version"
```

- Per i dettagli sull'API, vedere [UntagResources](#) in AWS Tools for PowerShell Cmdlet Reference.

Esempi di Route 53 che utilizzano Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with Route 53.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Edit-R53ResourceRecordSet

Il seguente esempio di codice mostra come utilizzare `Edit-R53ResourceRecordSet`.

Strumenti per PowerShell

Esempio 1: questo esempio crea un record A per `www.example.com` e modifica il record A per `test.example.com` da `192.0.2.3` a `192.0.2.1`. Nota che i valori per le modifiche ai record di tipo TXT devono essere racchiusi tra virgolette doppie. Per ulteriori dettagli, consulta la documentazione di Amazon Route 53. Puoi utilizzare il `Get-R53Change` cmdlet per effettuare un sondaggio per determinare quando le modifiche sono state completate.

```
$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
```

Esempio 2: questo esempio mostra come creare set di record di risorse alias. 'Z222222222' è l'ID della zona ospitata di Amazon Route 53 in cui stai creando il set di record di risorse alias. 'example.com' è l'apice della zona per cui desideri creare un alias e 'www.example.com' è un sottodominio per il quale desideri creare anche un alias. 'Z11111' è un esempio di ID di zona ospitata per il sistema di bilanciamento del carico e 'example-load-balancer-11.us-east-1.elb.amazonaws.com' è un esempio di nome di dominio del sistema di bilanciamento del carico con cui Amazon Route 53 risponde alle domande relative a example.com e www.example.com. Per ulteriori dettagli, consulta la documentazione di Amazon Route 53. Puoi utilizzare il Get-R53Change cmdlet per effettuare un sondaggio per determinare quando le modifiche sono state completate.

```

$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 = "Z111111111111111"
$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 = "Z111111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $false

$params = @{
    HostedZoneId="Z2222222222"
    ChangeBatch_Comment="This change batch creates two alias resource record sets, one
for the zone apex, example.com, and one for www.example.com, that both point to
example-load-balancer-1111111111.us-east-1.elb.amazonaws.com."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params

```

Esempio 3: questo esempio crea due record A per `www.example.com`. Un quarto delle volte (1/(1+3)), Amazon Route 53 risponde alle domande relative a `www.example.com` con i due valori del primo set di record di risorse (192.0.2.9 e 192.0.2.10). Tre quarti delle volte (3/(1+3)) Amazon Route 53 risponde alle domande relative a `www.example.com` con i due valori per il secondo set di record di risorse (192.0.2.11 e 192.0.2.12). Per ulteriori dettagli, consulta la documentazione di Amazon Route 53. Puoi utilizzare il `Get-R53Change` cmdlet per effettuare un sondaggio per determinare quando le modifiche sono state completate.

```

$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
```

Esempio 4: Questo esempio mostra come creare set di record di risorse alias ponderati presupponendo che `example.com` sia il dominio per il quale si desidera creare set di record di risorse alias ponderati. `SetIdentifier` differenzia i due set di record di risorse alias ponderati l'uno dall'altro. Questo elemento è obbligatorio perché gli elementi `Name` e `Type` hanno gli stessi valori per entrambi i set di record di risorse. `Z11111` e `Z33333` sono esempi di ID di zona ospitata per il sistema di bilanciamento del carico ELB specificato dal valore di `DNSName`. `example-load-balancer-2222222222.us-east-1.elb.amazonaws.com` e `example-load-balancer-4444444444.us-east-1.elb.amazonaws.com` sono esempi di domini Elastic Load Balancing da cui Amazon Route 53 risponde alle domande per `example.com`. Per ulteriori dettagli, consulta la documentazione di Amazon Route 53. Puoi utilizzare il `Get-R53Change` cmdlet per effettuare un sondaggio per determinare quando le modifiche sono state completate.

```
$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 = "Z11111111111111"
$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 = "Z33333333333333"
$change2.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-4444444444.us-east-1.elb.amazonaws.com."
$change2.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $false

$params = @{
    HostedZoneId="Z555555555555"
    ChangeBatch_Comment="This change batch creates two weighted alias resource
record sets. Amazon Route 53 responds to queries for example.com with the first ELB
domain 3/4ths of the times and the second one 1/4th of the time."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

Esempio 5: Questo esempio crea due set di record di risorse con alias di latenza, uno per un sistema di bilanciamento del carico ELB nella regione Stati Uniti occidentali (Oregon) (us-west-2) e un altro per un sistema di bilanciamento del carico nella regione Asia Pacifico (Singapore) (ap-southeast-1). Per ulteriori dettagli, consulta la documentazione di Amazon Route 53. Puoi utilizzare il `Get-R53Change` cmdlet per effettuare un sondaggio per determinare quando le modifiche sono state completate.

```

$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 = "Z111111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-222222222.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 = "Z222222222222222"
$change2.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.ap-southeast-1.elb.amazonaws.com"
$change2.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$params = @{
    HostedZoneId="Z555555555555"
    ChangeBatch_Comment="This change batch creates two latency resource record
sets, one for the US West (Oregon) region and one for the Asia Pacific (Singapore)
region."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params

```

- Per i dettagli sull'API, vedere [ChangeResourceRecordSets](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-R53AccountLimit

Il seguente esempio di codice mostra come utilizzare. `Get-R53AccountLimit`

Strumenti per PowerShell

Esempio 1: Questo esempio restituisce il numero massimo di zone ospitate che possono essere create utilizzando l'account corrente.

```
Get-R53AccountLimit -Type MAX_HOSTED_ZONES_BY_OWNER
```

Output:

```
15
```

- Per i dettagli sull'API, vedere [GetAccountLimit](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-R53CheckerIpRanges

Il seguente esempio di codice mostra come utilizzare `Get-R53CheckerIpRanges`

Strumenti per PowerShell

Esempio 1: questo esempio restituisce i CIDR per i controllori sanitari di Route53

```
Get-R53CheckerIpRanges
```

Output:

```
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
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [GetCheckerIpRanges](#) AWS Tools for PowerShell

Get-R53HostedZone

Il seguente esempio di codice mostra come utilizzare. `Get-R53HostedZone`

Strumenti per PowerShell

Esempio 1: restituisce i dettagli della zona ospitata con ID Z1D633PJN98FT9.

```
Get-R53HostedZone -Id Z1D633PJN98FT9
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [GetHostedZone](#) AWS Tools for PowerShell

Get-R53HostedZoneCount

Il seguente esempio di codice mostra come utilizzare. `Get-R53HostedZoneCount`

Strumenti per PowerShell

Esempio 1: restituisce il numero totale di zone ospitate pubbliche e private per l'attuale Account AWS.

```
Get-R53HostedZoneCount
```

- Per i dettagli sull'API, vedere [GetHostedZoneCount](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-R53HostedZoneLimit

Il seguente esempio di codice mostra come utilizzare. `Get-R53HostedZoneLimit`

Strumenti per PowerShell

Esempio 1: Questo esempio restituisce il limite al numero massimo di record che possono essere creati nella zona ospitata specificata.

```
Get-R53HostedZoneLimit -HostedZoneId Z3MEQ8T7HAAAAF -Type MAX_RRSETS_BY_ZONE
```

Output:

```
5
```

- Per i dettagli sull'API, vedere [GetHostedZoneLimit](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-R53HostedZoneList

Il seguente esempio di codice mostra come utilizzare. `Get-R53HostedZoneList`

Strumenti per PowerShell

Esempio 1: emette tutte le zone ospitate pubbliche e private.

```
Get-R53HostedZoneList
```

Esempio 2: restituisce tutte le zone ospitate associate al set di delega riutilizzabile con ID NZ8X2CISAMPLE

```
Get-R53HostedZoneList -DelegationSetId NZ8X2CISAMPLE
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [ListHostedZones](#) AWS Tools for PowerShell

Get-R53HostedZonesByName

Il seguente esempio di codice mostra come utilizzare. `Get-R53HostedZonesByName`

Strumenti per PowerShell

Esempio 1: restituisce tutte le zone ospitate pubbliche e private in ordine ASCII per nome di dominio.

```
Get-R53HostedZonesByName
```

Esempio 2: restituisce le zone ospitate pubbliche e private, in ordine ASCII per nome di dominio, a partire dal nome DNS specificato.

```
Get-R53HostedZonesByName -DnsName example2.com
```

Esempio 3: Questo esempio mostra come enumerare manualmente le zone ospitate recuperando prima un singolo elemento e poi iterandone due alla volta fino alla restituzione di tutte le zone, utilizzando le proprietà dei marker allegate alla risposta del servizio nello stack dopo ogni chiamata. **\$AWSHistory**

```
Get-R53HostedZonesByName -MaxItem 1
while ($LastServiceResponse.IsTruncated)
{
    $nextPageParams = @{
        DnsName=$LastServiceResponse.NextDNSName
        HostedZoneId=$LastServiceResponse.NextHostedZoneId
    }
    Get-R53HostedZonesByName -MaxItem 2 @nextPageParams
}
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [ListHostedZonesByName](#) AWS Tools for PowerShell

Get-R53QueryLoggingConfigList

Il seguente esempio di codice mostra come utilizzare. `Get-R53QueryLoggingConfigList`

Strumenti per PowerShell

Esempio 1: Questo esempio restituisce tutte le configurazioni per la registrazione delle query DNS associate alla versione corrente. Account AWS

```
Get-R53QueryLoggingConfigList
```

Output:

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:*

- Per i dettagli sull'API, vedere [ListQueryLoggingConfigs](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-R53ReusableDelegationSet

Il seguente esempio di codice mostra come utilizzare. `Get-R53ReusableDelegationSet`

Strumenti per PowerShell

Esempio 1: Questo esempio recupera le informazioni sul set di delega specificato, inclusi i quattro name server assegnati al set di delega.

```
Get-R53ReusableDelegationSet -Id N23DS9X4AYEAAA
```

Output:

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}

- Per i dettagli sull'API, vedere [GetReusableDelegationSet](#) in AWS Tools for PowerShell Cmdlet Reference.

New-R53HostedZone

Il seguente esempio di codice mostra come utilizzare. `New-R53HostedZone`

Strumenti per PowerShell

Esempio 1: crea una nuova zona ospitata denominata 'example.com', associata a un set di deleghe riutilizzabile. Si noti che è necessario fornire un valore per il `CallerReference` parametro in modo che le richieste in questione debbano essere ritentate, se necessario, senza il rischio di eseguire l'operazione due volte. Poiché la zona ospitata viene creata in un VPC, è automaticamente privata e non è necessario impostare il parametro `- HostedZoneConfig _PrivateZone`.

```
$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
```

- Per i dettagli sull'API, vedere [CreateHostedZone](#) in AWS Tools for PowerShell Cmdlet Reference.

New-R53QueryLoggingConfig

Il seguente esempio di codice mostra come utilizzare. `New-R53QueryLoggingConfig`

Strumenti per PowerShell

Esempio 1: Questo esempio crea una nuova configurazione di registrazione delle query DNS di Route53 per la zona ospitata specificata. Amazon Route53 pubblicherà i log delle query DNS nel gruppo di log Cloudwatch specificato.

```
New-R53QueryLoggingConfig -HostedZoneId Z3MEQ8T7HAAAAF -CloudWatchLogsLogGroupArn
arn:aws:logs:us-east-1:111111111111:log-group:/aws/route53/example.com:*
```

Output:

QueryLoggingConfig	Location
-----	-----
Amazon.Route53.Model.QueryLoggingConfig	https://route53.amazonaws.com/2013-04-01/queryloggingconfig/ee5aaa95-4e03-4fdc-9d28-9e24ddaaaaa3

- Per i dettagli sull'API, consulta Cmdlet Reference. [CreateQueryLoggingConfig](#) AWS Tools for PowerShell

New-R53ReusableDelegationSet

Il seguente esempio di codice mostra come utilizzare. `New-R53ReusableDelegationSet`

Strumenti per PowerShell

Esempio 1: Questo esempio crea un set di delega riutilizzabile di 4 name server che possono essere riutilizzati da più zone ospitate.

```
New-R53ReusableDelegationSet -CallerReference testcallerreference
```

Output:

DelegationSet	Location
-----	-----
Amazon.Route53.Model.DelegationSet	https://route53.amazonaws.com/2013-04-01/delegationset/N23DS9XAAAAAXM

- Per i dettagli sull'API, vedere [CreateReusableDelegationSet](#) in AWS Tools for PowerShell Cmdlet Reference.

Register-R53VPCWithHostedZone

Il seguente esempio di codice mostra come utilizzare. `Register-R53VPCWithHostedZone`

Strumenti per PowerShell

Esempio 1: questo esempio associa il VPC specificato alla zona ospitata privata.

```
Register-R53VPCWithHostedZone -HostedZoneId Z3MEQ8T7HAAAAF -VPC_VPCId vpc-f1b9aaaa -VPC_VPCRegion us-east-1
```

Output:

Id	Status	SubmittedAt	Comment
--	-----	-----	-----
/change/C3SCAAA633Z6DX	PENDING	01/28/2020 19:32:02	

- Per i dettagli sull'API, vedere [WithHostedZoneAssociateVPC](#) in Cmdlet Reference.AWS Tools for PowerShell

Remove-R53HostedZone

Il seguente esempio di codice mostra come utilizzare. Remove-R53HostedZone

Strumenti per PowerShell

Esempio 1: elimina la zona ospitata con l'ID specificato. Ti verrà richiesta una conferma prima di procedere con il comando, a meno che tu non aggiunga il parametro -Force switch.

```
Remove-R53HostedZone -Id Z1PA6795UKMFR9
```

- Per i dettagli sull'API, vedere [DeleteHostedZone](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-R53QueryLoggingConfig

Il seguente esempio di codice mostra come utilizzare. Remove-R53QueryLoggingConfig

Strumenti per PowerShell

Esempio 1: Questo esempio rimuove la configurazione specificata per la registrazione delle query DNS.

```
Remove-R53QueryLoggingConfig -Id ee528e95-4e03-4fdc-9d28-9e24daaa20063
```

- Per i dettagli sull'API, vedere [DeleteQueryLoggingConfig](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-R53ReusableDelegationSet

Il seguente esempio di codice mostra come utilizzare. Remove-R53ReusableDelegationSet

Strumenti per PowerShell

Esempio 1: Questo esempio elimina il set di deleghe riutilizzabile specificato.

```
Remove-R53ReusableDelegationSet -Id N23DS9X4AYAAAM
```

- Per i dettagli sull'API, vedere [DeleteReusableDelegationSet](#) in AWS Tools for PowerShell Cmdlet Reference.

Unregister-R53VPCFromHostedZone

Il seguente esempio di codice mostra come utilizzare. `Unregister-R53VPCFromHostedZone`

Strumenti per PowerShell

Esempio 1: questo esempio dissocia il VPC specificato dalla zona ospitata privata.

```
Unregister-R53VPCFromHostedZone -HostedZoneId Z3MEQ8T7HAAAAF -VPC_VPCId vpc-f1b9aaaa  
-VPC_VPCRegion us-east-1
```

Output:

Id	Status	SubmittedAt	Comment
--	-----	-----	-----
/change/C2XFCAAAA9HKZG	PENDING	01/28/2020 10:35:55	

- Per i dettagli sull'API, vedere [FromHostedZoneDisassociateVPC](#) in Cmdlet Reference.AWS Tools for PowerShell

Update-R53HostedZoneComment

Il seguente esempio di codice mostra come utilizzare. `Update-R53HostedZoneComment`

Strumenti per PowerShell

Esempio 1: questo comando aggiorna il commento per la zona ospitata specificata.

```
Update-R53HostedZoneComment -Id Z385PDS6AAAAAR -Comment "This is my first hosted  
zone"
```

Output:

```
Id           : /hostedzone/Z385PDS6AAAAAR
Name        : example.com.
CallerReference : C5B55555-7147-EF04-8341-69131E805C89
Config      : Amazon.Route53.Model.HostedZoneConfig
ResourceRecordSetCount : 9
LinkedService :
```

- Per i dettagli sull'API, vedere [UpdateHostedZoneComment](#) in AWS Tools for PowerShell Cmdlet Reference.

Esempi di Amazon S3 con Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon S3. AWS Tools for PowerShell

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Copy-S3Object

Il seguente esempio di codice mostra come utilizzare `Copy-S3Object`.

Strumenti per PowerShell

Esempio 1: Questo comando copia l'oggetto "sample.txt" dal bucket «test-files» allo stesso bucket ma con una nuova chiave "sample-copy.txt».

```
Copy-S3Object -BucketName test-files -Key sample.txt -DestinationKey sample-copy.txt
```

Esempio 2: Questo comando copia l'oggetto "sample.txt" dal bucket «test-files» al bucket «backup-files» con una chiave "sample-copy.txt».

```
Copy-S3Object -BucketName test-files -Key sample.txt -DestinationKey sample-copy.txt  
-DestinationBucket backup-files
```

Esempio 3: Questo comando scarica l'oggetto "sample.txt" dal bucket «test-files» in un file locale con nome "local-sample.txt».

```
Copy-S3Object -BucketName test-files -Key sample.txt -LocalFile local-sample.txt
```

Esempio 4: scarica il singolo oggetto nel file specificato. Il file scaricato si trova in c:\downloads\data\archive.zip

```
Copy-S3Object -BucketName test-files -Key data/archive.zip -LocalFolder c:\downloads
```

Esempio 5: scarica tutti gli oggetti che corrispondono al prefisso chiave specificato nella cartella locale. La gerarchia delle chiavi relativa verrà conservata come sottocartelle nella posizione generale di download.

```
Copy-S3Object -BucketName test-files -KeyPrefix data -LocalFolder c:\downloads
```

- Per i dettagli sull'API, vedere [CopyObject](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-S3ACL

Il seguente esempio di codice mostra come utilizzare. Get-S3ACL

Strumenti per PowerShell

Esempio 1: il comando ottiene i dettagli del proprietario dell'oggetto S3.

```
Get-S3ACL -BucketName 's3casetestbucket' -key 'initialize.ps1' -Select  
AccessControlList.Owner
```

Output:

```
DisplayName Id
----- --
testusername      9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544
```

- Per i dettagli sull'API, vedere [GetACL](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-S3Bucket

Il seguente esempio di codice mostra come utilizzare. `Get-S3Bucket`

Strumenti per PowerShell

Esempio 1: questo comando restituisce tutti i bucket S3.

```
Get-S3Bucket
```

Esempio 2: questo comando restituisce un bucket denominato «test-files»

```
Get-S3Bucket -BucketName test-files
```

- Per i dettagli sull'API, vedere [ListBuckets](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-S3BucketAccelerateConfiguration

Il seguente esempio di codice mostra come utilizzare. `Get-S3BucketAccelerateConfiguration`

Strumenti per PowerShell

Esempio 1: questo comando restituisce il valore `Enabled`, se le impostazioni di accelerazione del trasferimento sono abilitate per il bucket specificato.

```
Get-S3BucketAccelerateConfiguration -BucketName 's3testbucket'
```

Output:

```
Value
-----
```

Enabled

- Per i dettagli sull'API, vedere [GetBucketAccelerateConfiguration](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-S3BucketAnalyticsConfiguration

Il seguente esempio di codice mostra come utilizzare. `Get-S3BucketAnalyticsConfiguration`

Strumenti per PowerShell

Esempio 1: questo comando restituisce i dettagli del filtro di analisi con il nome 'testfilter' nel bucket S3 specificato.

```
Get-S3BucketAnalyticsConfiguration -BucketName 's3testbucket' -AnalyticsId  
'testfilter'
```

- Per i dettagli sull'API, vedere [GetBucketAnalyticsConfiguration](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-S3BucketAnalyticsConfigurationList

Il seguente esempio di codice mostra come utilizzare. `Get-S3BucketAnalyticsConfigurationList`

Strumenti per PowerShell

Esempio 1: questo comando restituisce le prime 100 configurazioni di analisi del bucket S3 specificato.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 's3casetestbucket'
```

- Per i dettagli sull'API, vedere [ListBucketAnalyticsConfigurations](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-S3BucketEncryption

Il seguente esempio di codice mostra come utilizzare. `Get-S3BucketEncryption`

Strumenti per PowerShell

Esempio 1: questo comando restituisce tutte le regole di crittografia lato server associate al bucket specificato.

```
Get-S3BucketEncryption -BucketName 's3casetestbucket'
```

- Per i dettagli sull'API, vedere [GetBucketEncryption](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-S3BucketInventoryConfiguration

Il seguente esempio di codice mostra come utilizzare. `Get-S3BucketInventoryConfiguration`

Strumenti per PowerShell

Esempio 1: questo comando restituisce i dettagli dell'inventario denominato 'testinventory' per il bucket S3 specificato.

```
Get-S3BucketInventoryConfiguration -BucketName 's3testbucket' -InventoryId  
'testinventory'
```

- Per i dettagli sull'API, vedere [GetBucketInventoryConfiguration](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-S3BucketInventoryConfigurationList

Il seguente esempio di codice mostra come utilizzare. `Get-S3BucketInventoryConfigurationList`

Strumenti per PowerShell

Esempio 1: questo comando restituisce le prime 100 configurazioni di inventario del bucket S3 specificato.

```
Get-S3BucketInventoryConfigurationList -BucketName 's3testbucket'
```

- Per i dettagli sull'API, vedere [ListBucketInventoryConfigurations](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-S3BucketLocation

Il seguente esempio di codice mostra come utilizzare. `Get-S3BucketLocation`

Strumenti per PowerShell

Esempio 1: questo comando restituisce il vincolo di posizione per il bucket 's3testbucket', se esiste un vincolo.

```
Get-S3BucketLocation -BucketName 's3testbucket'
```

Output:

```
Value
-----
ap-south-1
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [GetBucketLocation](#) AWS Tools for PowerShell

Get-S3BucketLogging

Il seguente esempio di codice mostra come utilizzare. `Get-S3BucketLogging`

Strumenti per PowerShell

Esempio 1: questo comando restituisce lo stato di registrazione per il bucket specificato.

```
Get-S3BucketLogging -BucketName 's3testbucket'
```

Output:

```
TargetBucketName  Grants TargetPrefix
-----
testbucket1       {}      testprefix
```

- Per i dettagli sull'API, vedere [GetBucketLogging](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-S3BucketMetricsConfiguration

Il seguente esempio di codice mostra come utilizzare. `Get-S3BucketMetricsConfiguration`

Strumenti per PowerShell

Esempio 1: questo comando restituisce i dettagli sul filtro metrico denominato 'testfilter' per il bucket S3 specificato.

```
Get-S3BucketMetricsConfiguration -BucketName 's3testbucket' -MetricsId 'testfilter'
```

- Per i dettagli sull'API, vedere [GetBucketMetricsConfiguration](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-S3BucketNotification

Il seguente esempio di codice mostra come utilizzare. `Get-S3BucketNotification`

Strumenti per PowerShell

Esempio 1: Questo esempio recupera la configurazione di notifica del bucket specificato

```
Get-S3BucketNotification -BucketName kt-tools | select -ExpandProperty  
TopicConfigurations
```

Output:

```
Id    Topic  
--    -  
mimo  arn:aws:sns:eu-west-1:123456789012:topic-1
```

- Per i dettagli sull'API, vedere [GetBucketNotification](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-S3BucketPolicy

Il seguente esempio di codice mostra come utilizzare. `Get-S3BucketPolicy`

Strumenti per PowerShell

Esempio 1: questo comando restituisce la policy del bucket associata al bucket S3 specificato.

```
Get-S3BucketPolicy -BucketName 's3testbucket'
```

- Per i dettagli sull'API, vedere [GetBucketPolicy](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-S3BucketPolicyStatus

Il seguente esempio di codice mostra come utilizzare. `Get-S3BucketPolicyStatus`

Strumenti per PowerShell

Esempio 1: questo comando restituisce lo stato della politica per il bucket S3 specificato, indicando se il bucket è pubblico.

```
Get-S3BucketPolicyStatus -BucketName 's3casetestbucket'
```

- Per i dettagli sull'API, vedere [GetBucketPolicyStatus](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-S3BucketReplication

Il seguente esempio di codice mostra come utilizzare. `Get-S3BucketReplication`

Strumenti per PowerShell

Esempio 1: restituisce le informazioni di configurazione della replica impostate nel bucket denominato 'mybucket'.

```
Get-S3BucketReplication -BucketName mybucket
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [GetBucketReplication](#) AWS Tools for PowerShell

Get-S3BucketRequestPayment

Il seguente esempio di codice mostra come utilizzare. `Get-S3BucketRequestPayment`

Strumenti per PowerShell

Esempio 1: restituisce la configurazione di pagamento della richiesta per il bucket denominato 'mybucket'. Per impostazione predefinita, il proprietario del bucket paga per i download dal bucket.

```
Get-S3BucketRequestPayment -BucketName mybucket
```

- Per i dettagli sull'API, vedere [GetBucketRequestPayment](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-S3BucketTagging

Il seguente esempio di codice mostra come utilizzare. `Get-S3BucketTagging`

Strumenti per PowerShell

Esempio 1: questo comando restituisce tutti i tag associati al bucket specificato.

```
Get-S3BucketTagging -BucketName 's3casetestbucket'
```

- Per i dettagli sull'API, vedere [GetBucketTagging](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-S3BucketVersioning

Il seguente esempio di codice mostra come utilizzare. `Get-S3BucketVersioning`

Strumenti per PowerShell

Esempio 1: questo comando restituisce lo stato del controllo delle versioni rispetto al bucket specificato.

```
Get-S3BucketVersioning -BucketName 's3testbucket'
```

- Per i dettagli sull'API, vedere [GetBucketVersioning](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-S3BucketWebsite

Il seguente esempio di codice mostra come utilizzare. `Get-S3BucketWebsite`

Strumenti per PowerShell

Esempio 1: questo comando restituisce i dettagli delle configurazioni statiche del sito Web del bucket S3 specificato.

```
Get-S3BucketWebsite -BucketName 's3testbucket'
```

- Per i dettagli sull'API, vedere [GetBucketWebsite](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-S3CORSConfiguration

Il seguente esempio di codice mostra come utilizzare. `Get-S3CORSConfiguration`

Strumenti per PowerShell

Esempio 1: questo comando restituisce un oggetto che contiene tutte le regole di configurazione CORS corrispondenti al bucket S3 specificato.

```
Get-S3CORSConfiguration -BucketName 's3testbucket' -Select Configuration.Rules
```

Output:

```
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 : {}
```

- Per i dettagli sull'API, vedere [GetCORSConfiguration](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-S3LifecycleConfiguration

Il seguente esempio di codice mostra come utilizzare `Get-S3LifecycleConfiguration`

Strumenti per PowerShell

Esempio 1: questo esempio recupera la configurazione del ciclo di vita per il bucket.

```
Get-S3LifecycleConfiguration -BucketName test-bla
```

Output:

```
Rules
-----
{Remove-in-150-days, Archive-to-Glacier-in-30-days}
```

- Per i dettagli sull'API, vedere [GetLifecycleConfiguration](#) in Cmdlet Reference. AWS Tools for PowerShell

Get-S3Object

Il seguente esempio di codice mostra come utilizzare `Get-S3Object`

Strumenti per PowerShell

Esempio 1: questo comando recupera le informazioni su tutti gli elementi nel bucket «test-files».

```
Get-S3Object -BucketName test-files
```

Esempio 2: questo comando recupera le informazioni sull'elemento "sample.txt" dal bucket «test-files».

```
Get-S3Object -BucketName test-files -Key sample.txt
```

Esempio 3: Questo comando recupera le informazioni su tutti gli elementi con il prefisso «sample» dal bucket «test-files».

```
Get-S3Object -BucketName test-files -KeyPrefix sample
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [ListObjects](#) AWS Tools for PowerShell

Get-S3ObjectLockConfiguration

Il seguente esempio di codice mostra come utilizzare. `Get-S3ObjectLockConfiguration`

Strumenti per PowerShell

Esempio 1: questo comando restituisce il valore 'Enabled' se la configurazione Object lock è abilitata per il bucket S3 specificato.

```
Get-S3ObjectLockConfiguration -BucketName 's3bucketteesting' -Select  
ObjectLockConfiguration.ObjectLockEnabled
```

Output:

```
Value  
-----  
Enabled
```

- Per i dettagli sull'API, vedere [GetObjectLockConfiguration](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-S3ObjectMetadata

Il seguente esempio di codice mostra come utilizzare. `Get-S3ObjectMetadata`

Strumenti per PowerShell

Esempio 1: questo comando restituisce i metadati dell'oggetto con la chiave 'ListTrusts.txt' nel bucket S3 specificato.

```
Get-S3ObjectMetadata -BucketName 's3testbucket' -Key 'ListTrusts.txt'
```

Output:

```
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               :

```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [GetObjectMetadata](#) AWS Tools for PowerShell

Get-S3ObjectRetention

Il seguente esempio di codice mostra come utilizzare. `Get-S3ObjectRetention`

Strumenti per PowerShell

Esempio 1: Il comando restituisce la modalità e la data fino a quando l'oggetto non viene mantenuto.

```
Get-S3ObjectRetention -BucketName 's3buckettesting' -Key 'testfile.txt'
```

- Per i dettagli sull'API, vedere [GetObjectRetention](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-S3ObjectTagSet

Il seguente esempio di codice mostra come utilizzare. `Get-S3ObjectTagSet`

Strumenti per PowerShell

Esempio 1: l'esempio restituisce i tag associati all'oggetto presente nel bucket S3 specificato.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'testbucket123'
```

Output:

```
Key Value
---
test value
```

- Per i dettagli sull'API, vedere [GetObjectTagging](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-S3PreSignedURL

Il seguente esempio di codice mostra come utilizzare. `Get-S3PreSignedURL`

Strumenti per PowerShell

Esempio 1: il comando restituisce un URL prefirmato per una chiave specificata e una data di scadenza.

```
Get-S3PreSignedURL -BucketName 's3testbucket' -Key 'testkey' -Expires '2023-11-16'
```

Esempio 2: il comando restituisce un URL prefirmato per un Directory Bucket con una chiave specificata e una data di scadenza.

```
[Amazon.AWSConfigsS3]::UseSignatureVersion4 = $true
Get-S3PreSignedURL -BucketName sampledirectorybucket--use1-az5--x-s3 -Key
'testkey' -Expire '2023-11-17'
```

- Per i dettagli sull'API, vedere [GetPreSignedURL](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-S3PublicAccessBlock

Il seguente esempio di codice mostra come utilizzare. `Get-S3PublicAccessBlock`

Strumenti per PowerShell

Esempio 1: il comando restituisce la configurazione del blocco di accesso pubblico del bucket S3 specificato.

```
Get-S3PublicAccessBlock -BucketName 's3testbucket'
```

- Per i dettagli sull'API, vedere [GetPublicAccessBlock](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-S3Version

Il seguente esempio di codice mostra come utilizzare. `Get-S3Version`

Strumenti per PowerShell

Esempio 1: questo comando restituisce i metadati relativi a tutte le versioni degli oggetti nel bucket S3 specificato.

```
Get-S3Version -BucketName 's3testbucket'
```

Output:

```
IsTruncated      : False
KeyMarker        :
VersionIdMarker  :
NextKeyMarker    :
NextVersionIdMarker :
Versions         : {EC2.txt, EC2MicrosoftWindowsGuide.txt, ListDirectories.json,
  ListTrusts.json}
Name             : s3testbucket
Prefix          :
MaxKeys         : 1000
CommonPrefixes  : {}
Delimiter       :
```

- Per i dettagli sull'API, vedere [ListVersions](#) in AWS Tools for PowerShell Cmdlet Reference.

New-S3Bucket

Il seguente esempio di codice mostra come utilizzare. `New-S3Bucket`

Strumenti per PowerShell

Esempio 1: questo comando crea un nuovo bucket privato denominato «sample-bucket».

```
New-S3Bucket -BucketName sample-bucket
```

Esempio 2: questo comando crea un nuovo bucket denominato «sample-bucket» con permessi di lettura-scrittura.

```
New-S3Bucket -BucketName sample-bucket -PublicReadWrite
```

Esempio 3: questo comando crea un nuovo bucket denominato «sample-bucket» con autorizzazioni di sola lettura.

```
New-S3Bucket -BucketName sample-bucket -PublicReadOnly
```

Esempio 4: questo comando crea un nuovo bucket di directory denominato «samplebucket--use1-az5--x-s3» con. 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
```

- Per i dettagli [PutBucket](#) sull'AWS Tools for PowerShell API, vedere in Cmdlet Reference.

Read-S3Object

Il seguente esempio di codice mostra come utilizzare. Read-S3Object

Strumenti per PowerShell

Esempio 1: Questo comando recupera l'elemento "sample.txt" dal bucket «test-files» e lo salva in un file denominato "local-sample.txt" nella posizione corrente. Il file "local-sample.txt" non deve esistere prima che questo comando venga chiamato.

```
Read-S3Object -BucketName test-files -Key sample.txt -File local-sample.txt
```

Esempio 2: Questo comando recupera la directory virtuale «DIR» dal bucket «test-files» e la salva in una cartella denominata «Local-dir» nella posizione corrente. La cartella «Local-dir» non deve esistere prima che questo comando venga chiamato.

```
Read-S3Object -BucketName test-files -KeyPrefix DIR -Folder Local-DIR
```

Esempio 3: scarica tutti gli oggetti con chiavi che terminano con '.json' dai bucket con 'config' nel nome del bucket ai file nella cartella specificata. Le chiavi degli oggetti vengono utilizzate per impostare i nomi dei file.

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

- Per i dettagli sull'API, vedere [GetObject](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-S3Bucket

Il seguente esempio di codice mostra come utilizzare. Remove-S3Bucket

Strumenti per PowerShell

Esempio 1: questo comando rimuove tutti gli oggetti e le versioni degli oggetti dal bucket 'test-files', quindi elimina il bucket. Il comando richiederà una conferma prima di procedere. Aggiungere l'interruttore -Force per sopprimere la conferma. Nota che i bucket che non sono vuoti non possono essere eliminati.

```
Remove-S3Bucket -BucketName test-files -DeleteBucketContent
```

- Per i dettagli sull'API, vedere [DeleteBucket](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-S3BucketAnalyticsConfiguration

Il seguente esempio di codice mostra come utilizzare. Remove-S3BucketAnalyticsConfiguration

Strumenti per PowerShell

Esempio 1: il comando rimuove il filtro di analisi con il nome 'testfilter' nel bucket S3 specificato.

```
Remove-S3BucketAnalyticsConfiguration -BucketName 's3testbucket' -AnalyticsId  
'testfilter'
```

- Per i dettagli sull'API, vedere [DeleteBucketAnalyticsConfiguration](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-S3BucketEncryption

Il seguente esempio di codice mostra come utilizzare. Remove-S3BucketEncryption

Strumenti per PowerShell

Esempio 1: questo disabilita la crittografia abilitata per il bucket S3 fornito.

```
Remove-S3BucketEncryption -BucketName 's3casetestbucket'
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DeleteBucketEncryption](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-S3BucketInventoryConfiguration

Il seguente esempio di codice mostra come utilizzare. Remove-S3BucketInventoryConfiguration

Strumenti per PowerShell

Esempio 1: questo comando rimuove l'inventario denominato 'testInventoryName' corrispondente al bucket S3 specificato.

```
Remove-S3BucketInventoryConfiguration -BucketName 's3testbucket' -InventoryId  
'testInventoryName'
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DeleteBucketInventoryConfiguration](#) in Cmdlet Reference.AWS Tools for PowerShell

Remove-S3BucketMetricsConfiguration

Il seguente esempio di codice mostra come utilizzare. `Remove-S3BucketMetricsConfiguration`

Strumenti per PowerShell

Esempio 1: il comando rimuove il filtro delle metriche con il nome 'testmetrics' nel bucket S3 specificato.

```
Remove-S3BucketMetricsConfiguration -BucketName 's3testbucket' -MetricsId  
'testmetrics'
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [DeleteBucketMetricsConfiguration](#) AWS Tools for PowerShell

Remove-S3BucketPolicy

Il seguente esempio di codice mostra come utilizzare. `Remove-S3BucketPolicy`

Strumenti per PowerShell

Esempio 1: il comando rimuove la policy del bucket associata al bucket S3 specificato.

```
Remove-S3BucketPolicy -BucketName 's3testbucket'
```

- Per i dettagli sull'API, vedere [DeleteBucketPolicy](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-S3BucketReplication

Il seguente esempio di codice mostra come utilizzare. Remove-S3BucketReplication

Strumenti per PowerShell

Esempio 1: elimina la configurazione di replica associata al bucket denominato 'mybucket'. Nota che questa operazione richiede l'autorizzazione per l'azione s3:. DeleteReplicationConfiguration Ti verrà richiesta una conferma prima che l'operazione prosegua. Per sopprimere la conferma, usa l'interruttore -Force.

```
Remove-S3BucketReplication -BucketName mybucket
```

- Per i dettagli sull'API, vedere [DeleteBucketReplication](#) in Cmdlet Reference.AWS Tools for PowerShell

Remove-S3BucketTagging

Il seguente esempio di codice mostra come utilizzare. Remove-S3BucketTagging

Strumenti per PowerShell

Esempio 1: questo comando rimuove tutti i tag associati al bucket S3 specificato.

```
Remove-S3BucketTagging -BucketName 's3testbucket'
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DeleteBucketTagging](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-S3BucketWebsite

Il seguente esempio di codice mostra come utilizzare `Remove-S3BucketWebsite`

Strumenti per PowerShell

Esempio 1: questo comando disabilita la proprietà statica di hosting del sito Web del bucket S3 specificato.

```
Remove-S3BucketWebsite -BucketName 's3testbucket'
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DeleteBucketWebsite](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-S3CORSConfiguration

Il seguente esempio di codice mostra come utilizzare `Remove-S3CORSConfiguration`

Strumenti per PowerShell

Esempio 1: questo comando rimuove la configurazione CORS per il bucket S3 specificato.

```
Remove-S3CORSConfiguration -BucketName 's3testbucket'
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DeleteCorsConfiguration in Cmdlet Reference](#).AWS Tools for PowerShell

Remove-S3LifecycleConfiguration

Il seguente esempio di codice mostra come utilizzare. Remove-S3LifecycleConfiguration
Strumenti per PowerShell

Esempio 1: il comando rimuove tutte le regole del ciclo di vita per il bucket S3 specificato.

```
Remove-S3LifecycleConfiguration -BucketName 's3testbucket'
```

- Per i dettagli sull'API, vedere [DeleteLifecycleConfiguration](#) in Cmdlet Reference.AWS Tools for PowerShell

Remove-S3MultipartUpload

Il seguente esempio di codice mostra come utilizzare. Remove-S3MultipartUpload
Strumenti per PowerShell

Esempio 1: questo comando interrompe i caricamenti in più parti creati prima di 5 giorni fa.

```
Remove-S3MultipartUpload -BucketName test-files -DaysBefore 5
```

Esempio 2: questo comando interrompe i caricamenti in più parti creati prima del 2 gennaio 2014.

```
Remove-S3MultipartUpload -BucketName test-files -InitiatedDate "Thursday, January  
02, 2014"
```

Esempio 3: questo comando interrompe i caricamenti in più parti creati prima del 2 gennaio 2014,
10:45:37.

```
Remove-S3MultipartUpload -BucketName test-files -InitiatedDate "2014/01/02 10:45:37"
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [AbortMultipartUpload](#) AWS Tools for PowerShell

Remove-S3Object

Il seguente esempio di codice mostra come utilizzare `Remove-S3Object`

Strumenti per PowerShell

Esempio 1: Questo comando rimuove l'oggetto "sample.txt" dal bucket «test-files». Viene richiesta una conferma prima dell'esecuzione del comando; per sopprimere il prompt, utilizzare l'opzione `-Force`.

```
Remove-S3Object -BucketName test-files -Key sample.txt
```

Esempio 2: questo comando rimuove la versione specificata dell'oggetto "sample.txt" dal bucket «test-files», presupponendo che il bucket sia stato configurato per abilitare le versioni degli oggetti.

```
Remove-S3Object -BucketName test-files -Key sample.txt -VersionId  
HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

Esempio 3: Questo comando rimuove gli oggetti "sample1.txt «," sample2.txt "e" sample3.txt "dal bucket «test-files» come un'unica operazione batch. La risposta del servizio elencherà tutte le chiavi elaborate, indipendentemente dallo stato di successo o di errore dell'eliminazione. Per ottenere solo gli errori relativi alle chiavi che non hanno potuto essere elaborate dal servizio, aggiungi il `ReportErrorsOnly` parametro - (questo parametro può essere specificato anche con l'alias `-Quiet`).

```
Remove-S3Object -BucketName test-files -KeyCollection @( "sample1.txt",  
"sample2.txt", "sample3.txt" )
```

Esempio 4: Questo esempio utilizza un'espressione in linea con il `KeyCollection` parametro - per ottenere le chiavi degli oggetti da eliminare. `Get-S3Object` restituisce una raccolta di istanze `Amazon.S3.Model.S3Object`, ognuna delle quali ha un membro `Key` di tipo string che identifica l'oggetto.

```
Remove-S3Object -bucketname "test-files" -KeyCollection (Get-S3Object "test-files" -  
KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

Esempio 5: Questo esempio ottiene tutti gli oggetti che hanno un prefisso chiave «prefix/subprefix» nel bucket e li elimina. Si noti che gli oggetti in entrata vengono elaborati uno alla volta.

Per raccolte di grandi dimensioni, è consigliabile passare la raccolta al parametro - InputObject (alias -S3ObjectCollection) del cmdlet per consentire l'eliminazione come batch con una singola chiamata al servizio.

```
Get-S3Object -BucketName "test-files" -KeyPrefix "prefix/subprefix" | Remove-S3Object -Force
```

Esempio 6: questo esempio reindirizza una raccolta di istanze Amazon.S3.Model.S3 ObjectVersion che rappresentano indicatori di eliminazione al cmdlet per l'eliminazione. Tieni presente che gli oggetti in entrata vengono elaborati uno alla volta. Per raccolte di grandi dimensioni, è consigliabile passare la raccolta al parametro - InputObject (alias -S3ObjectCollection) del cmdlet per consentire l'eliminazione come batch con una singola chiamata al servizio.

```
(Get-S3Version -BucketName "test-files").Versions | Where {$_.IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

Esempio 7: questo script mostra come eseguire un'eliminazione in batch di un set di oggetti (in questo caso eliminare i marker) creando una matrice di oggetti da utilizzare con il parametro - 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
```

- Per i dettagli sull'API, vedere [DeleteObjects](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-S3ObjectTagSet

Il seguente esempio di codice mostra come utilizzare Remove-S3ObjectTagSet

Strumenti per PowerShell

Esempio 1: questo comando rimuove tutti i tag associati all'oggetto con la chiave 'testfile.txt' nel bucket S3 specificato.

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 's3testbucket' -Select '^Key'
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DeleteObjectTagging](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-S3PublicAccessBlock

Il seguente esempio di codice mostra come utilizzare `Remove-S3PublicAccessBlock`

Strumenti per PowerShell

Esempio 1: questo comando disattiva l'impostazione di blocco dell'accesso pubblico per il bucket specificato.

```
Remove-S3PublicAccessBlock -BucketName 's3testbucket' -Force -Select '^BucketName'
```

Output:

```
s3testbucket
```

- Per i dettagli sull'API, vedere [DeletePublicAccessBlock](#) in AWS Tools for PowerShell Cmdlet Reference.

Set-S3BucketEncryption

Il seguente esempio di codice mostra come utilizzare `Set-S3BucketEncryption`

Strumenti per PowerShell

Esempio 1: questo comando abilita la crittografia AES256 lato server predefinita con Amazon S3 Managed Keys (SSE-S3) sul bucket specificato.

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =  
  @{{ServerSideEncryptionAlgorithm = "AES256"}}}  
Set-S3BucketEncryption -BucketName 's3testbucket' -  
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [PutBucketEncryption](#) AWS Tools for PowerShell

Test-S3Bucket

Il seguente esempio di codice mostra come utilizzare. Test-S3Bucket

Strumenti per PowerShell

Esempio 1: questo comando restituisce True se il bucket esiste, False in caso contrario. Il comando restituisce True anche se il bucket non appartiene all'utente.

```
Test-S3Bucket -BucketName test-files
```

- Per i dettagli sull'API, vedere [Test-S3Bucket](#) in AWS Tools for PowerShell Cmdlet Reference.

Write-S3BucketAccelerateConfiguration

Il seguente esempio di codice mostra come utilizzare. Write-S3BucketAccelerateConfiguration

Strumenti per PowerShell

Esempio 1: questo comando abilita l'accelerazione di trasferimento per il bucket S3 specificato.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')  
Write-S3BucketAccelerateConfiguration -BucketName 's3testbucket' -  
AccelerateConfiguration_Status $statusVal
```

- Per i dettagli sull'API, vedere [PutBucketAccelerateConfiguration](#) in AWS Tools for PowerShell Cmdlet Reference.

Write-S3BucketNotification

Il seguente esempio di codice mostra come utilizzare Write-S3BucketNotification

Strumenti per PowerShell

Esempio 1: questo esempio configura la configurazione dell'argomento SNS per l'evento S3 ObjectRemovedDelete e abilita la notifica per il bucket s3 specificato

```
$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
```

Esempio 2: questo esempio abilita le notifiche ObjectCreatedAll relative al bucket specificato che lo invia alla funzione Lambda.

```
$lambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = "s3:ObjectCreated:*"
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:rdplock"
    Id = "ObjectCreated-Lambda"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".pem"}
            )
        }
    }
}

Write-S3BucketNotification -BucketName ssm-editor -LambdaFunctionConfiguration
$lambdaConfig
```

Esempio 3: Questo esempio crea 2 diverse configurazioni Lambda sulla base di suffissi chiave diversi e configurate entrambe in un unico comando.

```
#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

```

- Per i dettagli sull'API, vedere [PutBucketNotification](#) in Cmdlet Reference.AWS Tools for PowerShell

Write-S3BucketReplication

Il seguente esempio di codice mostra come utilizzare Write-S3BucketReplication

Strumenti per PowerShell

Esempio 1: questo esempio imposta una configurazione di replica con un'unica regola che consente la replica nel bucket 'exampletargetbucket' di tutti i nuovi oggetti creati con il prefisso del nome chiave "" nel bucket 'examplebucket'. TaxDocs

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::exampletargetbucket" }

$params = @{
    BucketName = "examplebucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

Esempio 2: questo esempio imposta una configurazione di replica con più regole che consentono la replica nel bucket 'exampletargetbucket' di qualsiasi nuovo oggetto creato con il prefisso del nome chiave "TaxDocs" o OtherDocs ". I prefissi chiave non devono sovrapporsi.

```
$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
```

Esempio 3: questo esempio aggiorna la configurazione di replica nel bucket specificato per disabilitare la regola che controlla la replica degli oggetti con il prefisso del nome chiave "" nel bucket 'exampletargetbucket'. TaxDocs

```
$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
```

- Per [PutBucketReplication AWS Tools for PowerShell](#) i dettagli sull'API, vedere in Cmdlet Reference.

Write-S3BucketRequestPayment

Il seguente esempio di codice mostra come utilizzare Write-S3BucketRequestPayment

Strumenti per PowerShell

Esempio 1: aggiorna la configurazione del pagamento della richiesta per il bucket denominato «mybucket» in modo che alla persona che richiede i download dal bucket venga addebitato il costo del download. Per impostazione predefinita, il proprietario del bucket paga per i download. Per riportare il pagamento della richiesta ai valori predefiniti, usa 'BucketOwner' per il parametro RequestPaymentConfiguration_Payer.

```
Write-S3BucketRequestPayment -BucketName mybucket -RequestPaymentConfiguration_Payer  
Requester
```

- Per i dettagli sull'API, vedere [PutBucketRequestPayment](#) in AWS Tools for PowerShell Cmdlet Reference.

Write-S3BucketTagging

Il seguente esempio di codice mostra come utilizzare `Write-S3BucketTagging`

Strumenti per PowerShell

Esempio 1: questo comando applica due tag a un bucket denominato **cloudtrail-test-2018**: un tag con una chiave di Stage e un valore di Test e un tag con una chiave di Environment e un valore di Alpha. Per verificare che i tag siano stati aggiunti al bucket, esegui **Get-S3BucketTagging -BucketName bucket_name** I risultati dovrebbero mostrare i tag che hai applicato al bucket nel primo comando. Nota che **Write-S3BucketTagging** sovrascrive l'intero set di tag esistente su un bucket. Per aggiungere o eliminare singoli tag, eseguire i cmdlet Resource Groups and Tagging API e **Add-RGResourceTag Remove-RGResourceTag** In alternativa, utilizza Tag Editor nella Console di AWS gestione per gestire i tag bucket S3.

```
Write-S3BucketTagging -BucketName cloudtrail-test-2018 -TagSet @( @{ Key="Stage"; Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

Esempio 2: questo comando reindirizza un bucket denominato **cloudtrail-test-2018** nel cmdlet **Write-S3BucketTagging** Applica i tag Stage:Production e Department:Finance al bucket. Nota che **Write-S3BucketTagging** sovrascrive l'intero set di tag esistente su un bucket.

```
Get-S3Bucket -BucketName cloudtrail-test-2018 | Write-S3BucketTagging -TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department"; Value="Finance" } )
```

- Per i dettagli sull'API, vedere [PutBucketTagging](#) in AWS Tools for PowerShell Cmdlet Reference.

Write-S3BucketVersioning

Il seguente esempio di codice mostra come utilizzare `Write-S3BucketVersioning`

Strumenti per PowerShell

Esempio 1: il comando abilita il controllo delle versioni per il bucket S3 specificato.

```
Write-S3BucketVersioning -BucketName 's3testbucket' -VersioningConfig_Status Enabled
```

- Per i dettagli sull'API, vedere [PutBucketVersioning](#) in AWS Tools for PowerShell Cmdlet Reference.

Write-S3BucketWebsite

Il seguente esempio di codice mostra come utilizzare Write-S3BucketWebsite

Strumenti per PowerShell

Esempio 1: il comando abilita l'hosting del sito Web per il bucket specificato con il documento indice come 'index.html' e il documento di errore come 'error.html'.

```
Write-S3BucketWebsite -BucketName 's3testbucket' -  
WebsiteConfiguration_IndexDocumentSuffix 'index.html' -  
WebsiteConfiguration_ErrorDocument 'error.html'
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [PutBucketWebsite](#) AWS Tools for PowerShell

Write-S3LifecycleConfiguration

Il seguente esempio di codice mostra come utilizzare Write-S3LifecycleConfiguration

Strumenti per PowerShell

Esempio 1: Questo esempio scrive/sostituisce la configurazione fornita in \$NewRule. Questa configurazione assicura di limitare gli oggetti dell'ambito con determinati valori di prefisso e tag.

```
$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

```

Esempio 2: Questo esempio imposta più regole con il filtraggio. \$ ArchiveRule imposta gli oggetti da archiviare in 30 giorni su Glacier e 120 su. DeepArchive \$ ExpireRule scade sia la versione attuale che quella precedente in 150 giorni per gli oggetti con prefisso 'py' e tag:key 'archieved' impostato su 'yes'.

```

$ExpireRule = [Amazon.S3.Model.LifecycleRule] @{
    Expiration = @{
        Days= 150
    }
    Id = "Remove-in-150-days"
    Filter= @{
        LifecycleFilterPredicate = [Amazon.S3.Model.LifecycleAndOperator]@{
            Operands= @(
                [Amazon.S3.Model.LifecyclePrefixPredicate] @{
                    "Prefix" = "py"
                },
                [Amazon.S3.Model.LifecycleTagPredicate] @{
                    "Tag"= @{
                        "Key" = "archived"
                        "Value" = "yes"
                    }
                }
            )
        }
    }
}

```

```
Status= 'Enabled'
NoncurrentVersionExpiration = @{
    NoncurrentDays = 150
}
}

$ArchiveRule = [Amazon.S3.Model.LifecycleRule] @{
    Expiration = $null
    Id = "Archive-to-Glacier-in-30-days"
    Filter= @{
        LifecycleFilterPredicate = [Amazon.S3.Model.LifecycleAndOperator]@{
            Operands= @(
                [Amazon.S3.Model.LifecyclePrefixPredicate] @{
                    "Prefix" = "py"
                },
                [Amazon.S3.Model.LifecycleTagPredicate] @{
                    "Tag"= @{
                        "Key" = "reviewed"
                        "Value" = "yes"
                    }
                }
            )
        }
    )
}
Status = 'Enabled'
NoncurrentVersionExpiration = @{
    NoncurrentDays = 75
}
Transitions = @(
    @{
        Days = 30
        "StorageClass"= 'Glacier'
    },
    @{
        Days = 120
        "StorageClass"= [Amazon.S3.S3StorageClass]::DeepArchive
    }
)
}

Write-S3LifecycleConfiguration -BucketName my-review-scrap -Configuration_Rule
$ExpireRule,$ArchiveRule
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [PutLifecycleConfiguration](#) AWS Tools for PowerShell

Write-S3Object

Il seguente esempio di codice mostra come utilizzare `Write-S3Object`

Strumenti per PowerShell

Esempio 1: questo comando carica il singolo file "local-sample.txt" su Amazon S3, creando un oggetto con la chiave "sample.txt" nel bucket «test-files».

```
Write-S3Object -BucketName test-files -Key "sample.txt" -File .\local-sample.txt
```

Esempio 2: questo comando carica il singolo file "sample.txt" su Amazon S3, creando un oggetto con la chiave "sample.txt" nel bucket «test-files». Se il parametro `-Key` non viene fornito, il nome del file viene utilizzato come chiave dell'oggetto S3.

```
Write-S3Object -BucketName test-files -File .\sample.txt
```

Esempio 3: questo comando carica il singolo file "local-sample.txt" su Amazon S3, creando un oggetto con la chiave "prefix/to/sample.txt" nel bucket «test-files».

```
Write-S3Object -BucketName test-files -Key "prefix/to/sample.txt" -File .\local-sample.txt
```

Esempio 4: questo comando carica tutti i file nella sottodirectory «Scripts» nel bucket «test-files» e applica il prefisso chiave comune "" a ciascun oggetto. SampleScripts Ogni file caricato avrà una chiave "SampleScripts/filename" dove 'filename' varia.

```
Write-S3Object -BucketName test-files -Folder .\Scripts -KeyPrefix SampleScripts\
```

Esempio 5: questo comando carica tutti i file *.ps1 nella directory locale «Scripts» nel bucket «test-files» e applica il prefisso chiave comune "" a ciascun oggetto. SampleScripts Ogni file caricato avrà una chiave "/filename.ps1" dove 'filename' varia. SampleScripts

```
Write-S3Object -BucketName test-files -Folder .\Scripts -KeyPrefix SampleScripts\ -SearchPattern *.ps1
```

Esempio 6: questo comando crea un nuovo oggetto S3 contenente la stringa di contenuto specificata con la chiave 'sample.txt'.

```
Write-S3Object -BucketName test-files -Key "sample.txt" -Content "object contents"
```

Esempio 7: questo comando carica il file specificato (il nome del file viene utilizzato come chiave) e applica i tag specificati al nuovo oggetto.

```
Write-S3Object -BucketName test-files -File "sample.txt" -TagSet  
@{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

Esempio 8: questo comando carica in modo ricorsivo la cartella specificata e applica i tag specificati a tutti i nuovi oggetti.

```
Write-S3Object -BucketName test-files -Folder . -KeyPrefix "TaggedFiles" -Recurse -  
TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- Per i dettagli sull'API, vedere [PutObject](#) in AWS Tools for PowerShell Cmdlet Reference.

Write-S3ObjectRetention

Il seguente esempio di codice mostra come utilizzare `Write-S3ObjectRetention`

Strumenti per PowerShell

Esempio 1: il comando abilita la modalità di mantenimento della governance fino alla data '31 dicembre 2019 00:00:00' per l'oggetto 'testfile.txt' nel bucket S3 specificato.

```
Write-S3ObjectRetention -BucketName 's3bucketttesting' -Key 'testfile.txt' -  
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [PutObjectRetention](#) AWS Tools for PowerShell

Esempi di S3 Glacier che utilizzano Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with S3 Glacier.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Get-GLCJob

Il seguente esempio di codice mostra come utilizzare `Get-GLCJob`.

Strumenti per PowerShell

Esempio 1: restituisce i dettagli del lavoro specificato. Quando il processo viene completato correttamente, è possibile utilizzare il `JobOutput cmdlet Read-GC` per recuperare il contenuto del processo (un archivio o un elenco di inventario) nel file system locale.

```
Get-GLCJob -VaultName myvault -JobId "op1x...JSbthM"
```

Output:

```
Action                : ArchiveRetrieval
ArchiveId              : o909j...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

```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [DescribeJob](#) AWS Tools for PowerShell

New-GLCVault

Il seguente esempio di codice mostra come utilizzare. `New-GLCVault`

Strumenti per PowerShell

Esempio 1: crea un nuovo archivio per l'account dell'utente. Poiché non è stato fornito alcun valore al `AccountId` parametro `-`, i cmdlet utilizzano un valore predefinito di «-» che indica l'account corrente.

```
New-GLCVault -VaultName myvault
```

Output:

```
/01234567812/vaults/myvault
```

- Per i dettagli sull'API, vedere [CreateVault](#) in AWS Tools for PowerShell Cmdlet Reference.

Read-GLCJobOutput

Il seguente esempio di codice mostra come utilizzare. `Read-GLCJobOutput`

Strumenti per PowerShell

Esempio 1: scarica il contenuto dell'archivio il cui recupero era previsto nel processo specificato e lo archivia in un file su disco. Il download convalida automaticamente il checksum, se disponibile. Se necessario, il checksum può essere ottenuto dalla cronologia delle risposte del servizio in questo modo (supponendo che questo cmdlet sia stato l'ultima esecuzione):.

\$AWSHistory.LastServiceResponse Se il cmdlet non è stato eseguito più di recente, ispeziona la **\$AWSHistory.Commands** raccolta per ottenere la risposta del servizio pertinente.

```
Read-GLCJobOutput -VaultName myvault -JobId "HSWjArc...Zq2XLiW" -FilePath "c:\temp\blue.bin"
```

- Per i dettagli sull'API, vedere [GetJobOutput](#) in AWS Tools for PowerShell Cmdlet Reference.

Start-GLCJob

Il seguente esempio di codice mostra come utilizzare. `Start-GLCJob`

Strumenti per PowerShell

Esempio 1: avvia un processo per recuperare un archivio dal vault specificato di proprietà dell'utente. Lo stato del processo può essere verificato utilizzando il cmdlet `Get-GLCJob`. Quando il processo viene completato correttamente, è possibile utilizzare il `JobOutput` cmdlet `Read-GC` per recuperare il contenuto dell'archivio nel file system locale.

```
Start-GLCJob -VaultName myvault -JobType "archive-retrieval" -JobDescription "archive retrieval" -ArchiveId "o909j...TX-TpIhQJw"
```

Output:

JobId	JobOutputPath	Location
op1x...JSbthM		/012345678912/vaults/test/jobs/op1xe...I4HqCHKsJSbthM

- Per i dettagli sull'API, vedere in Cmdlet Reference. [InitiateJob](#) AWS Tools for PowerShell

Write-GLCArchive

Il seguente esempio di codice mostra come utilizzare. `Write-GLCArchive`

Strumenti per PowerShell

Esempio 1: carica un singolo file nel vault specificato, restituendo l'ID dell'archivio e il checksum calcolato.

```
Write-GLCArchive -VaultName myvault -FilePath c:\temp\blue.bin
```

Output:

FilePath	ArchiveId	Checksum
-----	-----	-----
C:\temp\blue.bin	o909jUUs...TTX-TpIhQJw	79f3e...f4395b

Esempio 2: carica il contenuto di una gerarchia di cartelle nell'archivio specificato nell'account dell'utente. Per ogni file caricato, il cmdlet emette il nome del file, l'ID di archivio corrispondente e il checksum calcolato dell'archivio.

```
Write-GLCArchive -VaultName myvault -FolderPath . -Recurse
```

Output:

FilePath	ArchiveId	Checksum
-----	-----	-----
C:\temp\blue.bin	o909jUUs...TTX-TpIhQJw	79f3e...f4395b
C:\temp\green.bin	qXAf0dSG...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	-Ix3jlm...iXiDh-Xf0PA	7469e...3e86f1

- Per i dettagli sull'API, vedere in Cmdlet Reference. [UploadArchive](#) AWS Tools for PowerShell

Esempi di Amazon SES con Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell con Amazon SES.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Get-SESIIdentity

Il seguente esempio di codice mostra come utilizzare `Get-SESIIdentity`.

Strumenti per PowerShell

Esempio 1: questo comando restituisce un elenco contenente tutte le identità (indirizzi e-mail e domini) per un AWS account specifico, indipendentemente dallo stato di verifica.

```
Get-SESIIdentity
```

- Per i dettagli sull'API, vedere [ListIdentities](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SESSendQuota

Il seguente esempio di codice mostra come utilizzare `Get-SESSendQuota`.

Strumenti per PowerShell

Esempio 1: questo comando restituisce i limiti di invio correnti dell'utente.

```
Get-SESSendQuota
```

- Per i dettagli sull'API, vedere [GetSendQuota](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SESSendStatistic

Il seguente esempio di codice mostra come utilizzare `Get-SESSendStatistic`.

Strumenti per PowerShell

Esempio 1: questo comando restituisce le statistiche di invio dell'utente. Il risultato è un elenco di punti dati, che rappresentano le ultime due settimane di attività di invio. Ogni punto dati nell'elenco contiene statistiche per un intervallo di 15 minuti.

```
Get-SESSendStatistic
```

- Per i dettagli sull'API, vedere [GetSendStatistics](#) in AWS Tools for PowerShell Cmdlet Reference.

Esempi di Amazon SNS con Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell con Amazon SNS.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Publish-SNSMessage

Il seguente esempio di codice mostra come utilizzare `Publish-SNSMessage`.

Strumenti per PowerShell

Esempio 1: Questo esempio mostra la pubblicazione di un messaggio con una sola riga `MessageAttribute` dichiarata.

```
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -Message  
"Hello" -MessageAttribute  
{City=[Amazon.SimpleNotificationService.Model.MessageAttributeValue]@{DataType='String';  
StringValue='AnyCity'}}
```

Esempio 2: Questo esempio mostra la pubblicazione di un messaggio con più messaggi `MessageAttributes` dichiarati in anticipo.

```
$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
```

- Per i dettagli sull'API, vedere [Publish](#) in AWS Tools for PowerShell Cmdlet Reference.

Esempi di Amazon SQS con Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell con Amazon SQS.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Add-SQSPermission

Il seguente esempio di codice mostra come utilizzare `Add-SQSPermission`.

Strumenti per PowerShell

Esempio 1: Questo esempio consente Account AWS all'utente specificato di inviare messaggi dalla coda specificata.

```
Add-SQSPermission -Action SendMessage -AWSAccountId 80398EXAMPLE -Label  
SendMessagesFromMyQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/  
MyQueue
```

- Per i dettagli sull'API, vedere [AddPermission](#) in AWS Tools for PowerShell Cmdlet Reference.

Clear-SQSQueue

Il seguente esempio di codice mostra come utilizzare `Clear-SQSQueue`.

Strumenti per PowerShell

Esempio 1: Questo esempio elimina tutti i messaggi dalla coda specificata.

```
Clear-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Per i dettagli sull'API, vedere [PurgeQueue](#) in AWS Tools for PowerShell Cmdlet Reference.

Edit-SQSMessageVisibility

Il seguente esempio di codice mostra come utilizzare `Edit-SQSMessageVisibility`.

Strumenti per PowerShell

Esempio 1: Questo esempio modifica il timeout di visibilità per il messaggio con l'handle di ricezione specificato nella coda specificata a 10 ore (10 ore* 60 minuti* 60 secondi = 36000 secondi).

```
Edit-SQSMessageVisibility -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/  
MyQueue -ReceiptHandle AQEBgGDh...J/Iqww== -VisibilityTimeout 36000
```


- Per i dettagli sull'API, vedere [ChangeMessageVisibility](#) in AWS Tools for PowerShell Cmdlet Reference.

Edit-SQSMessageVisibilityBatch

Il seguente esempio di codice mostra come utilizzare. Edit-SQSMessageVisibilityBatch

Strumenti per PowerShell

Esempio 1: Questo esempio modifica il timeout di visibilità per 2 messaggi con gli handle di ricezione specificati nella coda specificata. Il timeout di visibilità del primo messaggio viene modificato in 10 ore (10 ore* 60 minuti* 60 secondi = 36000 secondi). Il timeout di visibilità del secondo messaggio viene modificato in 5 ore (5 ore* 60 minuti* 60 secondi = 18000 secondi).

```
$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
```

Output:

```
Failed      Successful
-----
{}          {Request2, Request1}
```

- Per i dettagli sull'API, vedere [ChangeMessageVisibilityBatch](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SQSDeadLetterSourceQueue

Il seguente esempio di codice mostra come utilizzare. `Get-SQSDeadLetterSourceQueue`

Strumenti per PowerShell

Esempio 1: Questo esempio elenca gli URL di tutte le code che si basano sulla coda specificata come coda di lettere morte.

```
Get-SQSDeadLetterSourceQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

Output:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue  
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
```

- Per i dettagli sull'API, vedere [ListDeadLetterSourceQueues](#) in Cmdlet Reference. AWS Tools for PowerShell

Get-SQSQueue

Il seguente esempio di codice mostra come utilizzare. `Get-SQSQueue`

Strumenti per PowerShell

Esempio 1: questo esempio elenca tutte le code.

```
Get-SQSQueue
```

Output:

```
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
```

Esempio 2: questo esempio elenca tutte le code che iniziano con il nome specificato.

```
Get-SQSQueue -QueueNamePrefix My
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [ListQueues](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SQSQueueAttribute

Il seguente esempio di codice mostra come utilizzare `Get-SQSQueueAttribute`

Strumenti per PowerShell

Esempio 1: Questo esempio elenca tutti gli attributi per la coda specificata.

```
Get-SQSQueueAttribute -AttributeName All -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Output:

```
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":"Sid14495134224EX","Effect":"Allow","Principal":{"AWS":"*"},"Action":"SQS:SendMessage","Resource":"arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue","Condition":{"ArnEquals":{"aws:SourceArn":"arn:aws:sns:us-east-1:80398EXAMPLE:MyTopic"}}}, {"Sid":
```

```
"SendMessageFromMyQueue","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]...}
```

Esempio 2: Questo esempio elenca separatamente solo gli attributi specificati per la coda specificata.

```
Get-SQSQueueAttribute -AttributeName MaximumMessageSize, VisibilityTimeout -QueueUrl
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Output:

```
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":
    "SendMessageFromMyQueue","Effect":"Allow","Principal":
{"AWS":"80398EXAMPLE"},"Action":"SQS:SendMessage","Resource":
    arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue"}]}
```

```
Attributes : {[MaximumMessageSize, 262144],  
[VisibilityTimeout, 30]}
```

- Per i dettagli sull'API, vedere [GetQueueAttributes](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SQSQueueUrl

Il seguente esempio di codice mostra come utilizzare. `Get-SQSQueueUrl`

Strumenti per PowerShell

Esempio 1: questo esempio elenca l'URL della coda con il nome specificato.

```
Get-SQSQueueUrl -QueueName MyQueue
```

Output:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Per i dettagli sull'API, vedere [GetQueueUrl](#) in AWS Tools for PowerShell Cmdlet Reference.

New-SQSQueue

Il seguente esempio di codice mostra come utilizzare. `New-SQSQueue`

Strumenti per PowerShell

Esempio 1: Questo esempio crea una coda con il nome specificato.

```
New-SQSQueue -QueueName MyQueue
```

Output:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Per i dettagli sull'API, vedere [CreateQueue](#) in AWS Tools for PowerShell Cmdlet Reference.

Receive-SQSMessage

Il seguente esempio di codice mostra come utilizzare `Receive-SQSMessage`

Strumenti per PowerShell

Esempio 1: Questo esempio elenca le informazioni per un massimo di 10 messaggi da ricevere per la coda specificata. Le informazioni conterranno i valori per gli attributi del messaggio specificati, se esistono.

```
Receive-SQSMessage -AttributeName SenderId, SentTimestamp -MessageAttributeName
  StudentName, StudentGrade -MessageCount 10 -QueueUrl https://sqs.us-
  east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Output:

```
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==
```

- Per i dettagli sull'API, vedere [ReceiveMessage](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-SQSMessage

Il seguente esempio di codice mostra come utilizzare `Remove-SQSMessage`

Strumenti per PowerShell

Esempio 1: Questo esempio elimina il messaggio con l'handle di ricezione specificato dalla coda specificata.

```
Remove-SQSMessage -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
  -ReceiptHandle AQEBd329...v6gl8Q==
```

- Per i dettagli sull'API, vedere [DeleteMessage](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-SQSMessageBatch

Il seguente esempio di codice mostra come utilizzare. Remove-SQSMessageBatch

Strumenti per PowerShell

Esempio 1: Questo esempio elimina 2 messaggi con gli handle di ricezione specificati dalla coda specificata.

```
$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 = "AQEBq0VY...KTsLYg=="

Remove-SQSMessageBatch -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue -Entry $deleteMessageRequest1, $deleteMessageRequest2
```

Output:

```
Failed      Successful
-----
{}          {Request1, Request2}
```

- Per i dettagli sull'API, vedere [DeleteMessageBatch](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-SQSPermission

Il seguente esempio di codice mostra come utilizzare. Remove-SQSPermission

Strumenti per PowerShell

Esempio 1: Questo esempio rimuove le impostazioni di autorizzazione con l'etichetta specificata dalla coda specificata.

```
Remove-SQSPermission -Label SendMessageFromMyQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Per i dettagli sull'API, vedere [RemovePermission](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-SQSQueue

Il seguente esempio di codice mostra come utilizzare. Remove-SQSQueue

Strumenti per PowerShell

Esempio 1: questo esempio elimina la coda specificata.

```
Remove-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Per i dettagli sull'API, vedere [DeleteQueue](#) in AWS Tools for PowerShell Cmdlet Reference.

Send-SQSMessage

Il seguente esempio di codice mostra come utilizzare. Send-SQSMessage

Strumenti per PowerShell

Esempio 1: Questo esempio invia un messaggio con gli attributi e il corpo del messaggio specificati alla coda specificata con un ritardo di consegna del messaggio di 10 secondi.

```
$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
```

Output:

MD5ofMessageAttributes	MD5ofMessageBody	MessageId
-----	-----	-----
1d3e51347bc042efbdf6dda31EXAMPLE c739-4d0c-818b-1820eEXAMPLE	51b0a3256d59467f973009b73EXAMPLE	c35fed8f-

- Per i dettagli sull'API, vedere [SendMessage](#) in AWS Tools for PowerShell Cmdlet Reference.

Send-SQSMessageBatch

Il seguente esempio di codice mostra come utilizzare `Send-SQSMessageBatch`

Strumenti per PowerShell

Esempio 1: Questo esempio invia 2 messaggi con gli attributi e i corpi dei messaggi specificati alla coda specificata. La consegna viene ritardata di 15 secondi per il primo messaggio e di 10 secondi per il secondo messaggio.

```
$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

```

Output:

```

Failed      Successful
-----
{}          {FirstMessage, SecondMessage}

```

- Per i dettagli sull'API, vedere [SendMessageBatch](#) in AWS Tools for PowerShell Cmdlet Reference.

Set-SQSQueueAttribute

Il seguente esempio di codice mostra come utilizzare `Set-SQSQueueAttribute`

Strumenti per PowerShell

Esempio 1: Questo esempio mostra come impostare una policy che prevede la sottoscrizione di una coda a un argomento SNS. Quando un messaggio viene pubblicato sull'argomento, viene inviato un messaggio alla coda sottoscritta.

```

# 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 }
```

Esempio 2: Questo esempio imposta gli attributi specificati per la coda specificata.

```
Set-SQSQueueAttribute -Attribute @{"DelaySeconds" = "10"; "MaximumMessageSize" =
"131072"} -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Per i dettagli sull'API, vedere [SetQueueAttributes](#) in AWS Tools for PowerShell Cmdlet Reference.

AWS STS esempi di utilizzo di Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with AWS STS.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Convert-STSAuthorizationMessage

Il seguente esempio di codice mostra come utilizzare `Convert-STSAuthorizationMessage`.

Strumenti per PowerShell

Esempio 1: decodifica le informazioni aggiuntive contenute nel contenuto del messaggio codificato fornito e restituito in risposta a una richiesta. Le informazioni aggiuntive sono codificate perché i dettagli dello stato di autorizzazione possono costituire informazioni privilegiate che l'utente che ha richiesto l'azione non dovrebbe vedere.

```
Convert-STSAuthorizationMessage -EncodedMessage "...encoded message..."
```

- Per i dettagli sull'API, vedere [DecodeAuthorizationMessage](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-STSFederationToken

Il seguente esempio di codice mostra come utilizzare `Get-STSFederationToken`

Strumenti per PowerShell

Esempio 1: richiede un token federato valido per un'ora utilizzando «Bob» come nome dell'utente federato. Questo nome può essere usato per fare riferimento al nome utente federato in una

policy basata sulle risorse (ad esempio una bucket policy di Amazon S3). La policy IAM fornita, in formato JSON, viene utilizzata per definire le autorizzazioni disponibili per l'utente IAM. La policy fornita non può concedere più autorizzazioni di quelle concesse all'utente richiedente, e le autorizzazioni finali per l'utente federato sono il set più restrittivo in base all'intersezione tra la policy passata e la policy utente IAM.

```
Get-STSFederationToken -Name "Bob" -Policy "...JSON policy..." -DurationInSeconds
3600
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [GetFederationToken](#) AWS Tools for PowerShell

Get-STSSessionToken

Il seguente esempio di codice mostra come utilizzare. `Get-STSSessionToken`

Strumenti per PowerShell

Esempio 1: restituisce un'**Amazon.Runtime.AWSCredentials**istanza contenente credenziali temporanee valide per un determinato periodo di tempo. Le credenziali utilizzate per richiedere credenziali temporanee vengono dedotte dalle impostazioni predefinite correnti della shell. Per specificare altre credenziali, utilizzate i parametri `-o` / `-l`. `ProfileName` `AccessKey` `SecretKey`

```
Get-STSSessionToken
```

Output:

AccessKeyId	Expiration
SecretAccessKey	SessionToken
-----	-----
-----	-----
EXAMPLEACCESSKEYID	2/16/2015 9:12:28 PM
examplesecretaccesskey...	SamPleTokeN.....

Esempio 2: restituisce un'**Amazon.Runtime.AWSCredentials**istanza contenente credenziali temporanee valide per un'ora. Le credenziali utilizzate per effettuare la richiesta vengono ottenute dal profilo specificato.

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile
```

Output:

AccessKeyId	Expiration
SecretAccessKey	SessionToken
-----	-----
-----	-----
EXAMPLEACCESSKEYID	2/16/2015 9:12:28 PM
examplesecretaccesskey...	SamPleTokenN.....

Esempio 3: restituisce un'**Amazon.Runtime.AWSCredentials**istanza contenente credenziali temporanee valide per un'ora utilizzando il numero di identificazione del dispositivo MFA associato all'account le cui credenziali sono specificate nel profilo 'myprofile' e il valore fornito dal dispositivo.

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile -SerialNumber
YourMFADeviceSerialNumber -TokenCode 123456
```

Output:

AccessKeyId	Expiration
SecretAccessKey	SessionToken
-----	-----
-----	-----
EXAMPLEACCESSKEYID	2/16/2015 9:12:28 PM
examplesecretaccesskey...	SamPleTokenN.....

- Per i dettagli sull'API, vedere in Cmdlet Reference. [GetSessionToken](#) AWS Tools for PowerShell

Use-STSRole

Il seguente esempio di codice mostra come utilizzare. Use-STSRole

Strumenti per PowerShell

Esempio 1: restituisce un set di credenziali temporanee (chiave di accesso, chiave segreta e token di sessione) che possono essere utilizzate per un'ora per accedere a AWS risorse a cui l'utente richiedente potrebbe normalmente non avere accesso. Le credenziali restituite hanno le autorizzazioni consentite dalla politica di accesso del ruolo assunto e dalla politica fornita (non è possibile utilizzare la politica fornita per concedere autorizzazioni superiori a quelle definite dalla politica di accesso del ruolo assunto).

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -  
Policy "...JSON policy..." -DurationInSeconds 3600
```

Esempio 2: restituisce un set di credenziali temporanee, valide per un'ora, con le stesse autorizzazioni definite nella politica di accesso del ruolo assunto.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -  
DurationInSeconds 3600
```

Esempio 3: restituisce un set di credenziali temporanee che forniscono il numero di serie e il token generato da un MFA associato alle credenziali utente utilizzate per eseguire il cmdlet.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -  
DurationInSeconds 3600 -SerialNumber "GAHT12345678" -TokenCode "123456"
```

Esempio 4: restituisce un set di credenziali temporanee che hanno assunto un ruolo definito in un account cliente. Per ogni ruolo che la terza parte può assumere, l'account cliente deve creare un ruolo utilizzando un identificatore che deve essere passato nel ExternalId parametro - ogni volta che viene assunto il ruolo.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -  
DurationInSeconds 3600 -ExternalId "ABC123"
```

- Per i dettagli sull'API, vedere [AssumeRole](#) in AWS Tools for PowerShell Cmdlet Reference.

Use-STSWebIdentityRole

Il seguente esempio di codice mostra come utilizzare. Use-STSWebIdentityRole

Strumenti per PowerShell

Esempio 1: restituisce un set temporaneo di credenziali, valido per un'ora, per un utente che è stato autenticato con il provider di identità Login with Amazon. Le credenziali presuppongono la politica di accesso associata al ruolo identificato dal ruolo ARN. Facoltativamente, è possibile passare una policy JSON al parametro -Policy che perfeziona ulteriormente le autorizzazioni di accesso (non è possibile concedere più autorizzazioni di quelle disponibili nelle autorizzazioni associate al ruolo). Il valore fornito a - WebIdentityToken è l'identificatore utente univoco restituito dal provider di identità.

```
Use-STSWebIdentityRole -DurationInSeconds 3600 -ProviderId "www.amazon.com"  
-RoleSessionName "app1" -RoleArn "arn:aws:iam::123456789012:role/  
FederatedWebIdentityRole" -WebIdentityToken "Atza...DVI0r1"
```

- Per i dettagli sull'API, vedere [AssumeRoleWithWebIdentity](#) in AWS Tools for PowerShell Cmdlet Reference.

AWS Support esempi di utilizzo di Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with AWS Support.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Add-ASACommunicationToCase

Il seguente esempio di codice mostra come utilizzare `Add-ASACommunicationToCase`.

Strumenti per PowerShell

Esempio 1: aggiunge il corpo di una comunicazione e-mail al caso specificato.

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -  
CommunicationBody "Some text about the case"
```

Esempio 2: aggiunge il corpo di una comunicazione e-mail al caso specificato più uno o più indirizzi e-mail contenuti nella riga CC dell'e-mail.


```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -
CcEmailAddress @"(email1@address.com", "email2@address.com") -CommunicationBody
"Some text about the case"
```

- Per i dettagli sull'API, vedere [AddCommunicationToCase](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ASACase

Il seguente esempio di codice mostra come utilizzare. Get-ASACase

Strumenti per PowerShell

Esempio 1: restituisce i dettagli di tutti i casi di supporto.

```
Get-ASACase
```

Esempio 2: restituisce i dettagli di tutti i casi di supporto a partire dalla data e dall'ora specificate.

```
Get-ASACase -AfterTime "2013-09-10T03:06Z"
```

Esempio 3: restituisce i dettagli dei primi 10 casi di supporto, inclusi quelli che sono stati risolti.

```
Get-ASACase -MaxResult 10 -IncludeResolvedCases $true
```

Esempio 4: restituisce i dettagli del singolo caso di supporto specificato.

```
Get-ASACase -CaseIdList "case-12345678910-2013-c4c1d2bf33c5cf47"
```

Esempio 5: restituisce i dettagli dei casi di supporto specificati.

```
Get-ASACase -CaseIdList @"(case-12345678910-2013-c4c1d2bf33c5cf47",
"case-18929034710-2011-c4fdeabf33c5cf47")
```

Esempio 6: restituisce tutti i casi di supporto utilizzando il paging manuale. I casi vengono recuperati in lotti da 20.

```
$nextToken = $null
do {
```

```
Get-ASACase -NextToken $nextToken -MaxResult 20
$nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Per i dettagli sull'API, vedere [DescribeCases](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ASACommunication

Il seguente esempio di codice mostra come utilizzare. `Get-ASACommunication`

Strumenti per PowerShell

Esempio 1: restituisce tutte le comunicazioni per il caso specificato.

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

Esempio 2: restituisce tutte le comunicazioni dalla mezzanotte UTC del 1° gennaio 2012 per il caso specificato.

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -AfterTime
"2012-01-10T00:00Z"
```

Esempio 3: restituisce tutte le comunicazioni a partire dalla mezzanotte UTC del 1° gennaio 2012 per il caso specificato, utilizzando l'impaginazione manuale. Le comunicazioni vengono recuperate in batch da 20.

```
$nextToken = $null
do {
    Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -NextToken
    $nextToken -MaxResult 20
    $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Per i dettagli sull'API, vedere [DescribeCommunications](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ASAService

Il seguente esempio di codice mostra come utilizzare. `Get-ASAService`

Strumenti per PowerShell

Esempio 1: restituisce tutti i codici di servizio, i nomi e le categorie disponibili.

```
Get-ASAService
```

Esempio 2: restituisce il nome e le categorie del servizio con il codice specificato.

```
Get-ASAService -ServiceCodeList "amazon-cloudfront"
```

Esempio 3: restituisce il nome e le categorie per i codici di servizio specificati.

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch")
```

Esempio 4: restituisce il nome e le categorie (in giapponese) per i codici di servizio specificati. Attualmente sono supportati i codici di lingua inglese («en») e giapponese («ja»).

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch") -  
Language "ja"
```

- Per i dettagli sull'API, vedere [DescribeServices](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ASASeverityLevel

Il seguente esempio di codice mostra come utilizzare. Get-ASASeverityLevel

Strumenti per PowerShell

Esempio 1: restituisce l'elenco dei livelli di gravità che possono essere assegnati a un caso AWS Support.

```
Get-ASASeverityLevel
```

Esempio 2: restituisce l'elenco dei livelli di gravità che possono essere assegnati a un caso AWS Support. I nomi dei livelli vengono restituiti in giapponese.

```
Get-ASASeverityLevel -Language "ja"
```

- Per i dettagli sull'API, vedere [DescribeSeverityLevels](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ASATrustedAdvisorCheck

Il seguente esempio di codice mostra come utilizzare. `Get-ASATrustedAdvisorCheck`

Strumenti per PowerShell

Esempio 1: restituisce la raccolta di assegni Trusted Advisor. È necessario specificare il parametro `Language` che può accettare «en» per l'output in inglese o «ja» per l'output in giapponese.

```
Get-ASATrustedAdvisorCheck -Language "en"
```

- Per i dettagli sull'API, vedere [DescribeTrustedAdvisorChecks](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ASATrustedAdvisorCheckRefreshStatus

Il seguente esempio di codice mostra come utilizzare. `Get-ASATrustedAdvisorCheckRefreshStatus`

Strumenti per PowerShell

Esempio 1: restituisce lo stato corrente delle richieste di aggiornamento per i controlli specificati. `Request-ASA TrustedAdvisorCheckRefresh` può essere utilizzato per richiedere l'aggiornamento delle informazioni sullo stato dei controlli.

```
Get-ASATrustedAdvisorCheckRefreshStatus -CheckId @("checkid1", "checkid2")
```

- Per i dettagli sull'API, vedere [DescribeTrustedAdvisorCheckRefreshStatuses](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ASATrustedAdvisorCheckResult

Il seguente esempio di codice mostra come utilizzare. `Get-ASATrustedAdvisorCheckResult`

Strumenti per PowerShell

Esempio 1: restituisce i risultati di un controllo di Trusted Advisor. L'elenco dei controlli Trusted Advisor disponibili può essere ottenuto utilizzando `Get-ASATrustedAdvisorChecks`. L'output è

lo stato generale del controllo, il timestamp in cui il controllo è stato eseguito l'ultima volta e il checkid univoco per il controllo specifico. Per visualizzare i risultati in giapponese, aggiungi il parametro `-Language «ja»`.

```
Get-ASATrustedAdvisorCheckResult -CheckId "checkid1"
```

- Per i dettagli sull'API, vedere [DescribeTrustedAdvisorCheckResult](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-ASATrustedAdvisorCheckSummary

Il seguente esempio di codice mostra come utilizzare `Get-ASATrustedAdvisorCheckSummary`

Strumenti per PowerShell

Esempio 1: restituisce il riepilogo più recente per il controllo Trusted Advisor specificato.

```
Get-ASATrustedAdvisorCheckSummary -CheckId "checkid1"
```

Esempio 2: restituisce i riepiloghi più recenti per i controlli Trusted Advisor specificati.

```
Get-ASATrustedAdvisorCheckSummary -CheckId @("checkid1", "checkid2")
```

- Per i dettagli sull'API, vedere [DescribeTrustedAdvisorCheckSummaries](#) in AWS Tools for PowerShell Cmdlet Reference.

New-ASACase

Il seguente esempio di codice mostra come utilizzare `New-ASACase`

Strumenti per PowerShell

Esempio 1: crea un nuovo caso nel AWS Support Center. I valori per i `CategoryCode` parametri `-ServiceCode` e `-SeverityCode` possono essere ottenuti utilizzando il cmdlet `Get-AsaService`. Il valore per il `SeverityCode` parametro - può essere ottenuto utilizzando il cmdlet `Get-ASA`. `SeverityLevel` Il valore del `IssueType` parametro - può essere «assistenza clienti» o «tecnico». In caso di successo, viene emesso il numero del caso AWS Support. Per impostazione predefinita, il caso verrà gestito in inglese, per usare il giapponese aggiungi il parametro `-Language «ja»`. I `CommunicationBody` parametri `-ServiceCode`, `-CategoryCode`, `-Subject` e `-` sono obbligatori.

```
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"
```

- Per i dettagli sull'API, vedere [CreateCase](#) in AWS Tools for PowerShell Cmdlet Reference.

Request-ASATrustedAdvisorCheckRefresh

Il seguente esempio di codice mostra come utilizzare. Request-ASATrustedAdvisorCheckRefresh

Strumenti per PowerShell

Esempio 1: Richiede un aggiornamento per il controllo Trusted Advisor specificato.

```
Request-ASATrustedAdvisorCheckRefresh -CheckId "checkid1"
```

- Per i dettagli sull'API, vedere [RefreshTrustedAdvisorCheck](#) in AWS Tools for PowerShell Cmdlet Reference.

Resolve-ASACase

Il seguente esempio di codice mostra come utilizzare. Resolve-ASACase

Strumenti per PowerShell

Esempio 1: restituisce lo stato iniziale del caso specificato e lo stato corrente dopo il completamento della chiamata per risolverlo.

```
Resolve-ASACase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

- Per i dettagli sull'API, vedere [ResolveCase](#) in AWS Tools for PowerShell Cmdlet Reference.

Esempi di Systems Manager che utilizzano Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with Systems Manager.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Add-SSMResourceTag

Il seguente esempio di codice mostra come utilizzare `Add-SSMResourceTag`.

Strumenti per PowerShell

Esempio 1: questo esempio aggiorna una finestra di manutenzione con nuovi tag. Se il comando va a buon fine, non viene generato output. La sintassi utilizzata da questo esempio richiede la PowerShell versione 3 o successiva.

```
$option1 = @{Key="Stack";Value=@("Production")}
Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
  "MaintenanceWindow" -Tag $option1
```

Esempio 2: con PowerShell la versione 2, è necessario utilizzare `New-Object` per creare ogni tag. Non viene prodotto alcun risultato se il comando ha esito positivo.

```
$tag1 = New-Object Amazon.SimpleSystemsManagement.Model.Tag
$tag1.Key = "Stack"
$tag1.Value = "Production"

Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
  "MaintenanceWindow" -Tag $tag1
```

- Per i dettagli sull'API, vedere [AddTagsToResource](#) in AWS Tools for PowerShell Cmdlet Reference.

Edit-SSMDocumentPermission

Il seguente esempio di codice mostra come utilizzare. Edit-SSMDocumentPermission

Strumenti per PowerShell

Esempio 1: Questo esempio aggiunge le autorizzazioni di «condivisione» a tutti gli account di un documento. Non viene prodotto alcun risultato se il comando ha esito positivo.

```
Edit-SSMDocumentPermission -Name "RunShellScript" -PermissionType "Share" -
AccountIdsToAdd all
```

Esempio 2: questo esempio aggiunge le autorizzazioni di «condivisione» a un account specifico per un documento. Non viene emesso alcun risultato se il comando ha esito positivo.

```
Edit-SSMDocumentPermission -Name "RunShellScriptNew" -PermissionType "Share" -
AccountIdsToAdd "123456789012"
```

- Per i dettagli sull'API, vedere [ModifyDocumentPermission](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMActivation

Il seguente esempio di codice mostra come utilizzare. Get-SSMActivation

Strumenti per PowerShell

Esempio 1: questo esempio fornisce dettagli sulle attivazioni del tuo account.

```
Get-SSMActivation
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DescribeActivations](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMAssociation

Il seguente esempio di codice mostra come utilizzare. `Get-SSMAssociation`

Strumenti per PowerShell

Esempio 1: Questo esempio descrive l'associazione tra un'istanza e un documento.

```
Get-SSMAssociation -InstanceId "i-0000293ffd8c57862" -Name "AWS-UpdateSSMAgent"
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DescribeAssociation](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMAssociationExecution

Il seguente esempio di codice mostra come utilizzare. `Get-SSMAssociationExecution`

Strumenti per PowerShell

Esempio 1: questo esempio restituisce le esecuzioni per l'ID di associazione fornito

```
Get-SSMAssociationExecution -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

Output:

```

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

```

- Per i dettagli sull'API, vedere [DescribeAssociationExecutions](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMAssociationExecutionTarget

Il seguente esempio di codice mostra come utilizzare. Get-SSMAssociationExecutionTarget

Strumenti per PowerShell

Esempio 1: Questo esempio visualizza l'ID della risorsa e il relativo stato di esecuzione che fanno parte degli obiettivi di esecuzione dell'associazione

```

Get-SSMAssociationExecutionTarget -AssociationId 123a45a0-
c678-9012-3456-78901234db5e -ExecutionId 123a45a0-c678-9012-3456-78901234db5e |
Select-Object ResourceId, Status

```

Output:

```

ResourceId          Status
-----
i-0b1b2a3456f7a890b Success
i-01c12a45d6fc7a89f Success
i-0a1caf234f56d7dc8 Success
i-012a3fd45af6dbcfe Failed
i-0ddc1df23c4a5fb67 Success

```

Esempio 2: questo comando controlla la particolare esecuzione di una particolare automazione a partire da ieri, alla quale è associato un documento di comando. Controlla ulteriormente se l'esecuzione dell'associazione è fallita e, in tal caso, mostrerà i dettagli della chiamata del comando per l'esecuzione insieme all'ID dell'istanza

```
$AssociationExecution= Get-SSMAssociationExecutionTarget -
AssociationId 1c234567-890f-1aca-a234-5a678d901cb0 -ExecutionId
12345ca12-3456-2345-2b45-23456789012 |
    Where-Object {$_.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
    }
}
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DescribeAssociationExecutionTargets](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMAssociationList

Il seguente esempio di codice mostra come utilizzare. `Get-SSMAssociationList`

Strumenti per PowerShell

Esempio 1: questo esempio elenca tutte le associazioni per un'istanza. La sintassi utilizzata da questo esempio richiede la PowerShell versione 3 o successiva.

```
$filter1 = @{Key="InstanceId";Value=@"i-0000293ffd8c57862"}  
Get-SSMAssociationList -AssociationFilterList $filter1
```

Output:

```
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}
```

Esempio 2: questo esempio elenca tutte le associazioni per un documento di configurazione. La sintassi utilizzata da questo esempio richiede la PowerShell versione 3 o successiva.

```
$filter2 = @{Key="Name";Value=@"AWS-UpdateSSMAgent"}  
Get-SSMAssociationList -AssociationFilterList $filter2
```

Output:

```
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}
```

Esempio 3: con PowerShell la versione 2, è necessario utilizzare New-Object per creare ogni filtro.

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.AssociationFilter  
$filter1.Key = "InstanceId"
```

```
$filter1.Value = "i-0000293ffd8c57862"  
  
Get-SSMAssociationList -AssociationFilterList $filter1
```

Output:

```
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}
```

- Per i dettagli sull'API, vedere [ListAssociations](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMAssociationVersionList

Il seguente esempio di codice mostra come utilizzare `Get-SSMAssociationVersionList`

Strumenti per PowerShell

Esempio 1: questo esempio recupera tutte le versioni dell'associazione fornita.

```
Get-SSMAssociationVersionList -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

Output:

```
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}

```

- Per i dettagli sull'API, vedere [ListAssociationVersions](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMAutomationExecution

Il seguente esempio di codice mostra come utilizzare `Get-SSMAutomationExecution`

Strumenti per PowerShell

Esempio 1: Questo esempio visualizza i dettagli di un'esecuzione di automazione.

```

Get-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"

```

Output:

```

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_V2QyyN7NhSZQYpmVzpEc4oSnpj2GLTNYnXUHsTbqJkNMoDgubmbtthLmZyaiUYekORIrA42-
                             fv1x-04q5Fjff6glh

```

```

Yb6TI5b0GQeeNrpwNvpDzm0-
PSR1swlAbg9fdM9BcNjyrznsPukWpuKu9EC10u6v30XU1KC9nZ7mPlWMFZNkSioQqpWwEvMw-
GZktsQzm67q0hUhBNOLWYhbS
pkfiqzY-5nw3S0obx30fhd3EJa50_-
GjV_a0nFXQJa70ik40bF0rEh3MtCSbrQT6--DvFy_FQ8TKvkIXadyVskeJI84X0F5WmA60f1pi5GI08i-
nRfZS6oDeU
gELBjjoFKD8s3L2aI0B6umWVxnQ0jqhQRxwJ53b54sZJ2PW3v_mtg9-
q0CK0ezS3xfh_y0ilaUG0AZG-xjQFuvU_JZedWpla3xi-MZsmb1AifBI
(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...}

```

Esempio 2: questo esempio elenca i dettagli della fase per l'ID di esecuzione dell'automazione specificato

```

Get-SSMAutomationExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object -ExpandProperty StepExecutions | Select-Object
StepName, Action, StepStatus, ValidNextSteps

```

Output:

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	{}

- Per i dettagli sull'API, vedere [GetAutomationExecution](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMAutomationExecutionList

Il seguente esempio di codice mostra come utilizzare `Get-SSMAutomationExecutionList`

Strumenti per PowerShell

Esempio 1: Questo esempio descrive tutte le esecuzioni di automazione attive e terminate associate all'account.

```
Get-SSMAutomationExecutionList
```

Output:

```
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]]}
```

Esempio 2: questo esempio visualizza ExecutionID, document, timestamp di inizio/fine dell'esecuzione per le esecuzioni con valori diversi da «Success» AutomationExecutionStatus

```
Get-SSMAutomationExecutionList | Where-Object AutomationExecutionStatus
-ne "Success" | Select-Object AutomationExecutionId, DocumentName,
AutomationExecutionStatus, ExecutionStartTime, ExecutionEndTime | Format-Table -
AutoSize
```

Output:


```

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
                                4/16/2019 5:22:46 AM 4/16/2019 5:27:29 AM
                                Failed

```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [DescribeAutomationExecutions](#) AWS Tools for PowerShell

Get-SSMAutomationStepExecution

Il seguente esempio di codice mostra come utilizzare. `Get-SSMAutomationStepExecution`

Strumenti per PowerShell

Esempio 1: questo esempio visualizza informazioni su tutte le esecuzioni di fasi attive e terminate in un flusso di lavoro di automazione.

```

Get-SSMAutomationStepExecution -AutomationExecutionId e1d2bad3-4567-8901-ae23-456c7c8901be | Select-Object StepName, Action, StepStatus

```

Output:

```

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

- Per i dettagli sull'API, vedere [DescribeAutomationStepExecutions](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-SSMAvailablePatch

Il seguente esempio di codice mostra come utilizzare. Get-SSMAvailablePatch

Strumenti per PowerShell

Esempio 1: questo esempio ottiene tutte le patch disponibili per Windows Server 2012 che hanno una gravità MSRC pari a Critical. La sintassi utilizzata da questo esempio richiede la PowerShell versione 3 o successiva.

```
$filter1 = @{Key="PRODUCT";Values=@("WindowsServer2012")}
$filter2 = @{Key="MSRC_SEVERITY";Values=@("Critical")}

Get-SSMAvailablePatch -Filter $filter1,$filter2
```

Output:

```
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
...
```

Esempio 2: con PowerShell la versione 2, è necessario utilizzare New-Object per creare ogni filtro.

```
$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
```

Esempio 3: questo esempio recupera tutti gli aggiornamenti rilasciati negli ultimi 20 giorni e applicabili ai prodotti corrispondenti al 2019 WindowsServer

```
Get-SSMAvailablePatch | Where-Object ReleaseDate -ge (Get-Date).AddDays(-20) |
Where-Object Product -eq "WindowsServer2019" | Select-Object ReleaseDate, Product,
Title
```

Output:

```
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)
```

- Per i dettagli sull'API, vedere [DescribeAvailablePatches](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMCommand

Il seguente esempio di codice mostra come utilizzare. Get-SSMCommand

Strumenti per PowerShell

Esempio 1: questo esempio elenca tutti i comandi richiesti.

```
Get-SSMCommand
```

Output:

```
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       : {}
```

Esempio 2: Questo esempio ottiene lo stato di un comando specifico.

```
Get-SSMCommand -CommandId "4b75a163-d39a-4d97-87c9-98ae52c6be35"
```

Esempio 3: Questo esempio recupera tutti i comandi SSM richiamati dopo il 2019-04-01T00:00:00 Z

```
Get-SSMCommand -Filter @{Key="InvokedAfter";Value="2019-04-01T00:00:00Z"} | Select-Object CommandId, DocumentName, Status, RequestedDateTime | Sort-Object -Property RequestedDateTime -Descending
```

Output:

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

- Per AWS Tools for PowerShell i dettagli [ListCommands](#) sull'API, vedere in Cmdlet Reference.

Get-SSMCommandInvocation

Il seguente esempio di codice mostra come utilizzare `Get-SSMCommandInvocation`

Strumenti per PowerShell

Esempio 1: questo esempio elenca tutte le invocazioni di un comando.

```
Get-SSMCommandInvocation -CommandId "b8eac879-0541-439d-94ec-47a80d554f44" -Detail $true
```

Output:

```
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      :
```

Esempio 2: In questo esempio viene elencato l'ID di comando da invocare `CommandPlugins e1eb2e3c-ed4c-5123-45c1-234f5612345f`

```
Get-SSMCommandInvocation -CommandId e1eb2e3c-ed4c-5123-45c1-234f5612345f -Detail:
$true | Select-Object -ExpandProperty CommandPlugins
```

Output:

```
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/KUM0.png to ..\..\programdata\KUM0.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
```

- AWS Tools for PowerShell Per i dettagli [ListCommandInvocations](#)sull'API, vedere in Cmdlet Reference.

Get-SSMCommandInvocationDetail

Il seguente esempio di codice mostra come utilizzare `Get-SSMCommandInvocationDetail`

Strumenti per PowerShell

Esempio 1: Questo esempio visualizza i dettagli di un comando eseguito su un'istanza.

```
Get-SSMCommandInvocationDetail -InstanceId "i-0cb2b964d3e14fd9f" -CommandId
"b8eac879-0541-439d-94ec-47a80d554f44"
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [GetCommandInvocation](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMComplianceItemList

Il seguente esempio di codice mostra come utilizzare `Get-SSMComplianceItemList`

Strumenti per PowerShell

Esempio 1: questo esempio elenca l'elenco degli elementi di conformità per l'ID e il tipo di risorsa specificati, filtrando il tipo di conformità utilizzando «Associazione»

```
Get-SSMComplianceItemList -ResourceId i-1a2caf345f67d0dc2 -ResourceType
ManagedInstance -Filter @{Key="ComplianceType";Values="Association"}
```

Output:

```
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            :

```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [ListComplianceItems](#) AWS Tools for PowerShell

Get-SSMComplianceSummaryList

Il seguente esempio di codice mostra come utilizzare. `Get-SSMComplianceSummaryList`

Strumenti per PowerShell

Esempio 1: Questo esempio restituisce un conteggio riepilogativo delle risorse conformi e non conformi per tutti i tipi di conformità.

```
Get-SSMComplianceSummaryList
```

Output:

```

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

```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [ListComplianceSummaries](#) AWS Tools for PowerShell

Get-SSMConnectionStatus

Il seguente esempio di codice mostra come utilizzare. `Get-SSMConnectionStatus`

Strumenti per PowerShell

Esempio 1: Questo esempio recupera lo stato della connessione di Session Manager per un'istanza per determinare se è connessa e pronta a ricevere connessioni a Session Manager.

```
Get-SSMConnectionStatus -Target i-0a1caf234f12d3dc4
```

Output:

```
Status      Target
-----      -
Connected  i-0a1caf234f12d3dc4
```

- Per i dettagli sull'API, vedere [GetConnectionStatus](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMDefaultPatchBaseline

Il seguente esempio di codice mostra come utilizzare. `Get-SSMDefaultPatchBaseline`

Strumenti per PowerShell

Esempio 1: Questo esempio visualizza la linea di base della patch predefinita.

```
Get-SSMDefaultPatchBaseline
```

Output:

```
arn:aws:ssm:us-west-2:123456789012:patchbaseline/pb-04fb4ae6142167966
```

- Per i dettagli sull'API, vedere [GetDefaultPatchBaseline](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMDeployablePatchSnapshotForInstance

Il seguente esempio di codice mostra come utilizzare. Get-SSMDeployablePatchSnapshotForInstance

Strumenti per PowerShell

Esempio 1: questo esempio visualizza l'istantanea corrente per la linea di base della patch utilizzata da un'istanza. Questo comando deve essere eseguito dall'istanza utilizzando le credenziali dell'istanza. Per garantire che utilizzi le credenziali dell'istanza, l'esempio passa un **Amazon.Runtime.InstanceProfileAWSCredentials** oggetto al parametro Credentials.

```
$credentials = [Amazon.Runtime.InstanceProfileAWSCredentials]::new()
Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f" -Credentials $credentials
```

Output:

```
InstanceId          SnapshotDownloadUrl
-----
i-0cb2b964d3e14fd9f https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...1692/4681775b-098f-4435...
```

Esempio 2: questo esempio mostra come ottenere il valore completo. SnapshotDownloadUrl Questo comando deve essere eseguito dall'istanza utilizzando le credenziali dell'istanza. Per garantire che utilizzi le credenziali dell'istanza, l'esempio configura la PowerShell sessione per l'utilizzo di un oggetto. **Amazon.Runtime.InstanceProfileAWSCredentials**

```
Set-AWSCredential -Credential
([Amazon.Runtime.InstanceProfileAWSCredentials]::new())
(Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f").SnapshotDownloadUrl
```

Output:

```
https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...
```

- Per i dettagli sull'API, vedere [GetDeployablePatchSnapshotForInstance](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMDocument

Il seguente esempio di codice mostra come utilizzare. Get-SSMDocument

Strumenti per PowerShell

Esempio 1: questo esempio restituisce il contenuto di un documento.

```
Get-SSMDocument -Name "RunShellScript"
```

Output:

```
Content  
-----  
{...
```

Esempio 2: questo esempio visualizza il contenuto completo di un documento.

```
(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 }}"  
      }  
    }  
  ]  
}
```

```
}  
  }  
] }  
}
```

- Per i dettagli sull'API, vedere [GetDocument](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMDocumentDescription

Il seguente esempio di codice mostra come utilizzare. `Get-SSMDocumentDescription`

Strumenti per PowerShell

Esempio 1: questo esempio restituisce informazioni su un documento.

```
Get-SSMDocumentDescription -Name "RunShellScript"
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DescribeDocument](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMDocumentList

Il seguente esempio di codice mostra come utilizzare. `Get-SSMDocumentList`

Strumenti per PowerShell

Esempio 1: elenca tutti i documenti di configurazione del tuo account.

```
Get-SSMDocumentList
```

Output:

```
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
...
```

Esempio 2: Questo esempio recupera tutti i documenti di automazione il cui nome corrisponde a 'Platform'

```
Get-SSMDocumentList -DocumentFilterList @{Key="DocumentType";Value="Automation"} |
Where-Object Name -Match "Platform"
```

Output:

```
DocumentFormat    : JSON
DocumentType      : Automation
DocumentVersion   : 7
```

```
Name       : KT-Get-Platform
Owner      : 987654123456
PlatformTypes : {Windows, Linux}
SchemaVersion : 0.3
Tags       : {}
TargetType  :
VersionName :
```

- Per i dettagli sull'API, vedere [ListDocuments](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMDocumentPermission

Il seguente esempio di codice mostra come utilizzare `Get-SSMDocumentPermission`

Strumenti per PowerShell

Esempio 1: questo esempio elenca tutte le versioni di un documento.

```
Get-SSMDocumentVersionList -Name "RunShellScript"
```

Output:

CreatedDate	DocumentVersion	IsDefaultVersion	Name
2/24/2017 5:25:13 AM	1	True	RunShellScript

- Per i dettagli sull'API, vedere [DescribeDocumentPermission](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMDocumentVersionList

Il seguente esempio di codice mostra come utilizzare `Get-SSMDocumentVersionList`

Strumenti per PowerShell

Esempio 1: questo esempio restituisce l'elenco delle autorizzazioni per un documento.

```
Get-SSMDocumentPermission -Name "RunShellScript" -PermissionType "Share"
```

Output:

```
all
```

- Per i dettagli sull'API, vedere [ListDocumentVersions](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMEffectiveInstanceAssociationList

Il seguente esempio di codice mostra come utilizzare. Get-SSMEffectiveInstanceAssociationList

Strumenti per PowerShell

Esempio 1: Questo esempio descrive le associazioni efficaci per un'istanza.

```
Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -MaxResult 5
```

Output:

```
AssociationId          Content
-----
d8617c07-2079-4c18-9847-1655fc2698b0 {...
```

Esempio 2: questo esempio visualizza il contenuto delle associazioni efficaci per un'istanza.

```
(Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -MaxResult 5).Content
```

Output:

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

```

- Per i dettagli sull'API, vedere [DescribeEffectiveInstanceAssociations](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMEffectivePatchesForPatchBaseline

Il seguente esempio di codice mostra come utilizzare. Get-SSMEffectivePatchesForPatchBaseline

Strumenti per PowerShell

Esempio 1: questo esempio elenca tutte le linee di base delle patch, con un elenco di risultati massimo di 1.


```
Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1
```

Output:

```
Patch                                PatchStatus
-----                                -
Amazon.SimpleSystemsManagement.Model.Patch
Amazon.SimpleSystemsManagement.Model.PatchStatus
```

Esempio 2: questo esempio visualizza lo stato delle patch per tutte le patch di base, con un elenco di risultati massimo di 1.

```
(Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1).PatchStatus
```

Output:

```
ApprovalDate          DeploymentStatus
-----          -
12/21/2010 6:00:00 PM APPROVED
```

- Per i dettagli sull'API, vedere [DescribeEffectivePatchesForPatchBaseline](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMInstanceAssociationsStatus

Il seguente esempio di codice mostra come utilizzare `Get-SSMInstanceAssociationsStatus`

Strumenti per PowerShell

Esempio 1: questo esempio mostra i dettagli delle associazioni per un'istanza.

```
Get-SSMInstanceAssociationsStatus -InstanceId "i-0000293ffd8c57862"
```

Output:

```
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

```

Esempio 2: questo esempio controlla lo stato di associazione delle istanze per l'ID di istanza specificato e inoltre visualizza lo stato di esecuzione di tali associazioni

```

Get-SSMInstanceAssociationsStatus -InstanceId i-012e3cb4df567e8aa | ForEach-Object
{Get-SSMAssociationExecution -AssociationId .AssociationId}

```

Output:

```

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

```

- Per i dettagli sull'API, vedere [DescribeInstanceAssociationsStatus](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMInstanceInformation

Il seguente esempio di codice mostra come utilizzare. Get-SSMInstanceInformation

Strumenti per PowerShell

Esempio 1: questo esempio mostra i dettagli di ciascuna istanza.

```

Get-SSMInstanceInformation

```

Output:

```

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

```

Esempio 2: Questo esempio mostra come utilizzare il parametro `-Filter` per filtrare i risultati solo per le istanze di AWS Systems Manager nella regione **us-east-1** con un **AgentVersion** di **2.2.800.0**. È possibile trovare un elenco di valori di chiave `-Filter` validi nell'argomento di riferimento dell' InstanceInformation API (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

```

Output:

```

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

```

Esempio 3: Questo esempio mostra come utilizzare il `InstanceInformationFilterList` parametro - per filtrare i risultati solo in base alle istanze di AWS Systems Manager nella regione **us-east-1** con **PlatformTypes** of **Windows** o **Linux**. È possibile trovare un elenco di valori `InstanceInformationFilterList` chiave validi nell'argomento di riferimento dell'`InstanceInformationFilter` API (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

```

Output:

```

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
```

Esempio 4: questo esempio elenca le istanze e le esportazioni InstanceId gestite da ssm LastPingDateTime e in un file PlatformName csv. PingStatus

```
Get-SSMInstanceInformation | Select-Object InstanceId, PingStatus, LastPingDateTime,
PlatformName | Export-Csv Instance-details.csv -NoTypeInfo
```

- Per i dettagli sull'API, vedere [DescribeInstanceInformation](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMInstancePatch

Il seguente esempio di codice mostra come utilizzare. Get-SSMInstancePatch

Strumenti per PowerShell

Esempio 1: questo esempio ottiene i dettagli sulla conformità delle patch per un'istanza.

```
Get-SSMInstancePatch -InstanceId "i-08ee91c0b17045407"
```

- Per i dettagli sull'API, vedere [DescribeInstancePatches](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMInstancePatchState

Il seguente esempio di codice mostra come utilizzare. Get-SSMInstancePatchState

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene gli stati di riepilogo delle patch per un'istanza.

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407"
```

Esempio 2: Questo esempio ottiene gli stati di riepilogo delle patch per due istanze.

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407","i-09a618aec652973a9"
```

- Per i dettagli sull'API, vedere [DescribeInstancePatchStates](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMInstancePatchStatesForPatchGroup

Il seguente esempio di codice mostra come utilizzare. Get-SSMInstancePatchStatesForPatchGroup

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene gli stati di riepilogo delle patch per istanza per un gruppo di patch.

```
Get-SSMInstancePatchStatesForPatchGroup -PatchGroup "Production"
```

- Per i dettagli sull'API, vedere [DescribeInstancePatchStatesForPatchGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMInventory

Il seguente esempio di codice mostra come utilizzare. Get-SSMInventory

Strumenti per PowerShell

Esempio 1: questo esempio ottiene i metadati personalizzati per il tuo inventario.

```
Get-SSMInventory
```

Output:

```
Data
  Id
----
--
{[AWS:InstanceInformation,
 Amazon.SimpleSystemsManagement.Model.InventoryResultItem]} i-0cb2b964d3e14fd9f
```

- Per i dettagli sull'API, vedere [GetInventory](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMInventoryEntriesList

Il seguente esempio di codice mostra come utilizzare. Get-SSMInventoryEntriesList

Strumenti per PowerShell

Esempio 1: questo esempio elenca tutte le voci di inventario personalizzate per un'istanza.

```
Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
"Custom:RackInfo"
```

Output:

```
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
```

Esempio 2: questo esempio elenca i dettagli.

```
(Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
"Custom:RackInfo").Entries
```

Output:

```
Key           Value
---           -
RackLocation Bay B/Row C/Rack D/Shelf E
```

- Per i dettagli sull'API, vedere [ListInventoryEntries](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMInventorySchema

Il seguente esempio di codice mostra come utilizzare. Get-SSMInventorySchema

Strumenti per PowerShell

Esempio 1: questo esempio restituisce un elenco di nomi dei tipi di inventario per l'account.

```
Get-SSMInventorySchema
```

- Per i dettagli sull'API, vedere [GetInventorySchema](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMLatestEC2Image

Il seguente esempio di codice mostra come utilizzare. Get-SSMLatestEC2Image

Strumenti per PowerShell

Esempio 1: questo esempio elenca tutte le AMI Windows più recenti.

```
PS Get-SSMLatestEC2Image -Path ami-windows-latest
```

Output:

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	
...	

Esempio 2: questo esempio recupera l'ID AMI di un'immagine Amazon Linux specifica per la regione us-west-2.

```
PS Get-SSMLatestEC2Image -Path ami-amazon-linux-latest -ImageName amzn-ami-hvm-x86_64-ebs -Region us-west-2
```

Output:

```
ami-09b92cd132204c704
```

Esempio 3: questo esempio elenca tutte le AMI Windows più recenti che corrispondono all'espressione wildcard specificata.

```
Get-SSMLatestEC2Image -Path ami-windows-latest -ImageName *Windows*2019*English*
```

Output:

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
...	

- Per i dettagli sull'API, vedere [Get-SSMLatestec2Image](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-SSMMaintenanceWindow

Il seguente esempio di Get-SSMMaintenanceWindow codice mostra come utilizzare.

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene dettagli su una finestra di manutenzione.

```
Get-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d"
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [GetMaintenanceWindow](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMMaintenanceWindowExecution

Il seguente esempio di codice mostra come utilizzare. `Get-SSMMaintenanceWindowExecution`

Strumenti per PowerShell

Esempio 1: Questo esempio elenca le informazioni su un'attività eseguita come parte dell'esecuzione di una finestra di manutenzione.

```
Get-SSMMaintenanceWindowExecution -WindowExecutionId "518d5565-5969-4cca-8f0e-da3b2a638355"
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [GetMaintenanceWindowExecution](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMMaintenanceWindowExecutionList

Il seguente esempio di codice mostra come utilizzare. `Get-SSMMaintenanceWindowExecutionList`

Strumenti per PowerShell

Esempio 1: Questo esempio elenca tutte le esecuzioni per una finestra di manutenzione.

```
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d"
```

Output:

```
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
```

Esempio 2: Questo esempio elenca tutte le esecuzioni per una finestra di manutenzione prima di una data specificata.

```
$option1 = @{Key="ExecutedBefore";Values=@("2016-11-04T05:00:00Z")}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
$option1
```

Esempio 3: Questo esempio elenca tutte le esecuzioni per una finestra di manutenzione dopo una data specificata.

```
$option1 = @{Key="ExecutedAfter";Values=@("2016-11-04T05:00:00Z")}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
$option1
```

- Per i dettagli sull'API, vedere [DescribeMaintenanceWindowExecutions](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMMaintenanceWindowExecutionTask

Il seguente esempio di codice mostra come utilizzare. Get-SSMMaintenanceWindowExecutionTask

Strumenti per PowerShell

Esempio 1: Questo esempio elenca le informazioni su un'attività che faceva parte dell'esecuzione di una finestra di manutenzione.

```
Get-SSMMaintenanceWindowExecutionTask -TaskId "ac0c6ae1-daa3-4a89-832e-d384503b6586"
-WindowExecutionId "518d5565-5969-4cca-8f0e-da3b2a638355"
```

Output:

```

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.SimpleSystemsManag
    meterValueExpression]}
Type             : RUN_COMMAND
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355

```

- Per i dettagli sull'API, vedere [GetMaintenanceWindowExecutionTask](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMMaintenanceWindowExecutionTaskInvocationList

Il seguente esempio di codice mostra come utilizzare. `Get-SSMMaintenanceWindowExecutionTaskInvocationList`

Strumenti per PowerShell

Esempio 1: Questo esempio elenca le chiamate per un'attività eseguita come parte dell'esecuzione di una finestra di manutenzione.

```

Get-SSMMaintenanceWindowExecutionTaskInvocationList -TaskId "ac0c6ae1-
daa3-4a89-832e-d384503b6586" -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"

```

Output:

```

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  :
```

- Per i dettagli sull'API, vedere [DescribeMaintenanceWindowExecutionTaskInvocations](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMMaintenanceWindowExecutionTaskList

Il seguente esempio di codice mostra come utilizzare. `Get-SSMMaintenanceWindowExecutionTaskList`

Strumenti per PowerShell

Esempio 1: Questo esempio elenca le attività associate all'esecuzione di una finestra di manutenzione.

```
Get-SSMMaintenanceWindowExecutionTaskList -WindowExecutionId
"518d5565-5969-4cca-8f0e-da3b2a638355"
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DescribeMaintenanceWindowExecutionTasks](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMMaintenanceWindowList

Il seguente esempio di codice mostra come utilizzare. `Get-SSMMaintenanceWindowList`

Strumenti per PowerShell

Esempio 1: questo esempio elenca tutte le finestre di manutenzione del tuo account.

```
Get-SSMMaintenanceWindowList
```

Output:

```
Cutoff      : 1
Duration    : 4
Enabled     : True
Name        : My-First-Maintenance-Window
WindowId    : mw-06d59c1a07c022145
```

- Per i dettagli sull'API, vedere [DescribeMaintenanceWindows](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMMaintenanceWindowTarget

Il seguente esempio di codice mostra come utilizzare. Get-SSMMaintenanceWindowTarget

Strumenti per PowerShell

Esempio 1: Questo esempio elenca tutti gli obiettivi per una finestra di manutenzione.

```
Get-SSMMaintenanceWindowTarget -WindowId "mw-06cf17cbefcb4bf4f"
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DescribeMaintenanceWindowTargets](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMMaintenanceWindowTaskList

Il seguente esempio di codice mostra come utilizzare. Get-SSMMaintenanceWindowTaskList

Strumenti per PowerShell

Esempio 1: questo esempio elenca tutte le attività per una finestra di manutenzione.

```
Get-SSMMaintenanceWindowTaskList -WindowId "mw-06cf17cbefcb4bf4f"
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DescribeMaintenanceWindowTasks](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMParameterHistory

Il seguente esempio di codice mostra come utilizzare. Get-SSMParameterHistory

Strumenti per PowerShell

Esempio 1: questo esempio elenca la cronologia dei valori di un parametro.

```
Get-SSMParameterHistory -Name "Welcome"
```


Output:

```
Description      :  
KeyId            :  
LastModifiedDate : 3/3/2017 6:55:25 PM  
LastModifiedUser : arn:aws:iam::123456789012:user/admin  
Name             : Welcome  
Type             : String  
Value           : helloWorld
```

- Per i dettagli sull'API, vedere [GetParameterHistory](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMParameterList

Il seguente esempio di codice mostra come utilizzare. `Get-SSMParameterList`

Strumenti per PowerShell

Esempio 1: questo esempio elenca tutti i parametri.

```
Get-SSMParameterList
```

Output:

```
Description      :  
KeyId            :  
LastModifiedDate : 3/3/2017 6:58:23 PM  
LastModifiedUser : arn:aws:iam::123456789012:user/admin  
Name             : Welcome  
Type             : String
```

- Per i dettagli sull'API, vedere [DescribeParameters](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMParameterValue

Il seguente esempio di codice mostra come utilizzare. `Get-SSMParameterValue`

Strumenti per PowerShell

Esempio 1: questo esempio elenca i valori di un parametro.

```
Get-SSMParameterValue -Name "Welcome"
```

Output:

```
InvalidParameters Parameters
-----
{}                  {Welcome}
```

Esempio 2: questo esempio elenca i dettagli del valore.

```
(Get-SSMParameterValue -Name "Welcome").Parameters
```

Output:

```
Name      Type      Value
----      -
Welcome  String   Good day, Sunshine!
```

- Per i dettagli sull'API, vedere [GetParameters](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMPatchBaseline

Il seguente esempio di codice mostra come utilizzare `Get-SSMPatchBaseline`

Strumenti per PowerShell

Esempio 1: questo esempio elenca tutte le linee di base delle patch.

```
Get-SSMPatchBaseline
```

Output:

```
BaselineDescription
-----
BaselineName      BaselineId
-----
```

```

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...

```

Esempio 2: Questo esempio elenca tutte le linee di base delle patch fornite da AWS. La sintassi utilizzata da questo esempio richiede la PowerShell versione 3 o successiva.

```
$filter1 = @{"Key"="OWNER";Values=@("AWS")}
```

Output:

```
Get-SSMPatchBaseline -Filter $filter1
```

Esempio 3: questo esempio elenca tutte le linee di base delle patch con l'utente come proprietario. La sintassi utilizzata da questo esempio richiede la PowerShell versione 3 o successiva.

```
$filter1 = @{"Key"="OWNER";Values=@("Self")}
```

Output:

```
Get-SSMPatchBaseline -Filter $filter1
```

Esempio 4: con PowerShell la versione 2, è necessario utilizzare `New-Object` per creare ogni tag.

```

$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "OWNER"
$filter1.Values = "AWS"

Get-SSMPatchBaseline -Filter $filter1

```

Output:

```

BaselineDescription      BaselineId
                        BaselineName      DefaultBaselin
                        e

```

```
-----
-----
-----
Default Patch Baseline Provided by AWS. arn:aws:ssm:us-
west-2:123456789012:patchbaseline/pb-04fb4ae6142167966 AWS-DefaultPatchBaseline True
```

- Per i dettagli sull'API, vedere [DescribePatchBaselines](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMPatchBaselineDetail

Il seguente esempio di codice mostra come utilizzare. `Get-SSMPatchBaselineDetail`

Strumenti per PowerShell

Esempio 1: Questo esempio visualizza i dettagli di una patch di base.

```
Get-SSMPatchBaselineDetail -BaselineId "pb-03da896ca3b68b639"
```

Output:

```
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  : {}
```

- Per i dettagli sull'API, vedere [GetPatchBaseline](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMPatchBaselineForPatchGroup

Il seguente esempio di codice mostra come utilizzare. `Get-SSMPatchBaselineForPatchGroup`

Strumenti per PowerShell

Esempio 1: Questo esempio mostra la linea di base delle patch per un gruppo di patch.

```
Get-SSMPatchBaselineForPatchGroup -PatchGroup "Production"
```

Output:

```
BaselineId          PatchGroup
-----
pb-045f10b4f382baeda Production
```

- Per i dettagli sull'API, vedere [GetPatchBaselineForPatchGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMPatchGroup

Il seguente esempio di codice mostra come utilizzare. Get-SSMPatchGroup

Strumenti per PowerShell

Esempio 1: questo esempio elenca le registrazioni dei gruppi di patch.

```
Get-SSMPatchGroup
```

Output:

```
BaselineIdentity          PatchGroup
-----
Amazon.SimpleSystemsManagement.Model.PatchBaselineIdentity Production
```

- Per i dettagli sull'API, vedere [DescribePatchGroups](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMPatchGroupState

Il seguente esempio di codice mostra come utilizzare. Get-SSMPatchGroupState

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene il riepilogo di alto livello sulla conformità delle patch per un gruppo di patch.

```
Get-SSMPatchGroupState -PatchGroup "Production"
```

Output:

```
Instances : 4
InstancesWithFailedPatches : 1
InstancesWithInstalledOtherPatches : 4
InstancesWithInstalledPatches : 3
InstancesWithMissingPatches : 0
InstancesWithNotApplicablePatches : 0
```

- Per i dettagli sull'API, vedere [DescribePatchGroupState](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMResourceComplianceSummaryList

Il seguente esempio di codice mostra come utilizzare. `Get-SSMResourceComplianceSummaryList`

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene un conteggio riepilogativo a livello di risorsa. Il riepilogo include informazioni sullo stato conforme e non conforme e sui conteggi dettagliati di gravità degli elementi di conformità per i prodotti che corrispondono a «Windows10». Poiché l' `MaxResult` impostazione predefinita è 100 se il parametro non è specificato e questo valore non è valido, il parametro viene aggiunto e il valore è impostato su 50. `MaxResult`

```
$FilterValues = @{
    "Key"="Product"
    "Type"="EQUAL"
    "Values"="Windows10"
}

Get-SSMResourceComplianceSummaryList -Filter $FilterValues -MaxResult 50
```

- Per i dettagli sull'API, vedere [ListResourceComplianceSummaries](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-SSMResourceTag

Il seguente esempio di codice mostra come utilizzare. `Get-SSMResourceTag`

Strumenti per PowerShell

Esempio 1: questo esempio elenca i tag per una finestra di manutenzione.

```
Get-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
  "MaintenanceWindow"
```

Output:

```
Key      Value
---      -
Stack    Production
```

- Per i dettagli sull'API, vedere [ListTagsForResource](#) in AWS Tools for PowerShell Cmdlet Reference.

New-SSMActivation

Il seguente esempio di codice mostra come utilizzare `New-SSMActivation`

Strumenti per PowerShell

Esempio 1: questo esempio crea un'istanza gestita.

```
New-SSMActivation -DefaultInstanceName "MyWebServers" -IamRole "SSMAutomationRole" -
  RegistrationLimit 10
```

Output:

```
ActivationCode      ActivationId
-----
KWChh0xBTiwDcKE9B1KC 08e51e79-1e36-446c-8e63-9458569c1363
```

- Per i dettagli sull'API, vedere [CreateActivation](#) in AWS Tools for PowerShell Cmdlet Reference.

New-SSMAssociation

Il seguente esempio di codice mostra come utilizzare `New-SSMAssociation`

Strumenti per PowerShell

Esempio 1: Questo esempio associa un documento di configurazione a un'istanza, utilizzando gli ID delle istanze.

```
New-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

Output:

```
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 :
```

Esempio 2: questo esempio associa un documento di configurazione a un'istanza, utilizzando obiettivi.

```
$target = @{{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}}
New-SSMAssociation -Name "AWS-UpdateSSMAgent" -Target $target
```

Output:

```
Name           : AWS-UpdateSSMAgent
InstanceId      :
Date           : 3/1/2017 6:22:21 PM
Status.Name     :
Status.Date     :
Status.Message  :
Status.AdditionalInfo :
```

Esempio 3: Questo esempio associa un documento di configurazione a un'istanza, utilizzando destinazioni e parametri.

```
$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

```

Output:

```

Name                : Configure-CloudWatch
InstanceId           :
Date                : 5/17/2018 3:17:44 PM
Status.Name         :
Status.Date         :
Status.Message      :
Status.AdditionalInfo :

```

Esempio 4: Questo esempio crea un'associazione con tutte le istanze della regione, con. **AWS-GatherSoftwareInventory** Fornisce inoltre file personalizzati e posizioni di registro nei parametri da raccogliere

```

$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)"

```

Output:

```

Name                : AWS-GatherSoftwareInventory
InstanceId           :
Date                : 6/9/2019 8:57:56 AM
Status.Name         :
Status.Date         :
Status.Message      :
Status.AdditionalInfo :

```

- Per i dettagli sull'API, vedere [CreateAssociation](#) in AWS Tools for PowerShell Cmdlet Reference.

New-SSMAssociationFromBatch

Il seguente esempio di codice mostra come utilizzare. New-SSMAssociationFromBatch

Strumenti per PowerShell

Esempio 1: questo esempio associa un documento di configurazione a più istanze. L'output restituisce un elenco di operazioni riuscite e non riuscite, se applicabile.

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@"AWS-UpdateSSMAgent"}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@"AWS-UpdateSSMAgent"}
New-SSMAssociationFromBatch -Entry $option1,$option2
```

Output:

```
Failed    Successful
-----  -
{}        {Amazon.SimpleSystemsManagement.Model.FailedCreateAssociation,
Amazon.SimpleSystemsManagement.Model.FailedCreateAsso...
```

Esempio 2: Questo esempio mostrerà tutti i dettagli di un'operazione riuscita.

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@"AWS-UpdateSSMAgent"}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@"AWS-UpdateSSMAgent"}
(New-SSMAssociationFromBatch -Entry $option1,$option2).Successful
```

- Per i dettagli sull'API, vedere [CreateAssociationBatch](#) in AWS Tools for PowerShell Cmdlet Reference.

New-SSMDocument

Il seguente esempio di codice mostra come utilizzare. New-SSMDocument

Strumenti per PowerShell

Esempio 1: questo esempio crea un documento nel tuo account. Il documento deve essere in formato JSON. Per ulteriori informazioni sulla scrittura di un documento di configurazione, consulta il Documento di configurazione nel riferimento all'API SSM.

```
New-SSMDocument -Content (Get-Content -Raw "c:\temp\RunShellScript.json") -Name
"RunShellScript" -DocumentType "Command"
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [CreateDocument](#) in AWS Tools for PowerShell Cmdlet Reference.

New-SSMMaintenanceWindow

Il seguente esempio di codice mostra come utilizzare `New-SSMMaintenanceWindow`

Strumenti per PowerShell

Esempio 1: Questo esempio crea una nuova finestra di manutenzione con il nome specificato che viene eseguita alle 16:00 di ogni martedì per 4 ore, con un limite di 1 ora e che consente destinazioni non associate.

```
New-SSMMaintenanceWindow -Name "MyMaintenanceWindow" -Duration 4 -Cutoff 1 -
AllowUnassociatedTarget $true -Schedule "cron(0 16 ? * TUE *)"
```

Output:

```
mw-03eb53e1ea7383998
```

- Per i dettagli sull'API, vedere [CreateMaintenanceWindow](#) in AWS Tools for PowerShell Cmdlet Reference.

New-SSMPatchBaseline

Il seguente esempio di codice mostra come utilizzare `New-SSMPatchBaseline`

Strumenti per PowerShell

Esempio 1: questo esempio crea una linea di base delle patch che approva le patch, sette giorni dopo il loro rilascio da parte di Microsoft, per le istanze gestite che eseguono Windows Server 2019 in un ambiente di produzione.

```
$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
```

Output:

```
pb-0z4z6221c4296b23z
```

- Per i dettagli sull'API, vedere [CreatePatchBaseline](#) in Cmdlet Reference.AWS Tools for PowerShell

Register-SSMDefaultPatchBaseline

Il seguente esempio di codice mostra come utilizzare. Register-SSMDefaultPatchBaseline

Strumenti per PowerShell

Esempio 1: Questo esempio registra una patch di base come baseline di patch predefinita.

```
Register-SSMDefaultPatchBaseline -BaselineId "pb-03da896ca3b68b639"
```

Output:

```
pb-03da896ca3b68b639
```

- Per i dettagli sull'API, vedere [RegisterDefaultPatchBaseline](#) in AWS Tools for PowerShell Cmdlet Reference.

Register-SSMPatchBaselineForPatchGroup

Il seguente esempio di codice mostra come utilizzare. Register-SSMPatchBaselineForPatchGroup

Strumenti per PowerShell

Esempio 1: questo esempio registra una linea di base di patch per un gruppo di patch.

```
Register-SSMPatchBaselineForPatchGroup -BaselineId "pb-03da896ca3b68b639" -
PatchGroup "Production"
```

Output:

```
BaselineId          PatchGroup
-----
pb-03da896ca3b68b639 Production
```

- Per i dettagli sull'API, vedere [RegisterPatchBaselineForPatchGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

Register-SSMTargetWithMaintenanceWindow

Il seguente esempio di codice mostra come utilizzare. Register-SSMTargetWithMaintenanceWindow

Strumenti per PowerShell

Esempio 1: questo esempio registra un'istanza con una finestra di manutenzione.

```
$option1 = @{Key="InstanceIds";Values=@("i-0000293ffd8c57862")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
$option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

Output:

```
d8e47760-23ed-46a5-9f28-927337725398
```

Esempio 2: questo esempio registra più istanze con una finestra di manutenzione.

```
$option1 =
  @{Key="InstanceIds";Values=@("i-0000293ffd8c57862","i-0cb2b964d3e14fd9f")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
$option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

Output:

```
6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

Esempio 3: questo esempio registra un'istanza con una finestra di manutenzione utilizzando i tag EC2.

```
$option1 = @{Key="tag:Environment";Values=@("Production")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
$option1 -OwnerInformation "Production Web Servers" -ResourceType "INSTANCE"
```

Output:

```
2994977e-aefb-4a71-beac-df620352f184
```

- Per i dettagli sull'API, vedere [RegisterTargetWithMaintenanceWindow](#) in AWS Tools for PowerShell Cmdlet Reference.

Register-SSMTaskWithMaintenanceWindow

Il seguente esempio di codice mostra come utilizzare. Register-SSMTaskWithMaintenanceWindow

Strumenti per PowerShell

Esempio 1: questo esempio registra un'attività con una finestra di manutenzione utilizzando un ID di istanza. L'output è il 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
```

Output:

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

Esempio 2: questo esempio registra un'attività con una finestra di manutenzione utilizzando un ID di destinazione. L'output è il 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
```

Output:

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

Esempio 3: Questo esempio crea un oggetto parametro per il documento di comando run **AWS-RunPowerShellScript** e crea un'attività con una determinata finestra di manutenzione utilizzando l'ID di destinazione. L'output restituito è l'ID dell'attività.

```
$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
```

Output:

```
f1e2ef34-5678-12e3-456a-12334c5c6cbe
```

Esempio 4: Questo esempio registra un'attività di AWS Systems Manager Automation utilizzando un documento denominato **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"
```

- Per i dettagli sull'API, vedere [RegisterTaskWithMaintenanceWindow](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-SSMActivation

Il seguente esempio di codice mostra come utilizzare `Remove-SSMActivation`

Strumenti per PowerShell

Esempio 1: questo esempio elimina un'attivazione. Non viene emesso alcun output se il comando ha esito positivo.

```
Remove-SSMActivation -ActivationId "08e51e79-1e36-446c-8e63-9458569c1363"
```

- Per i dettagli sull'API, vedere [DeleteActivation](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-SSMAssociation

Il seguente esempio di codice mostra come utilizzare. Remove-SSMAssociation

Strumenti per PowerShell

Esempio 1: Questo esempio elimina l'associazione tra un'istanza e un documento. Se il comando ha esito positivo, non viene prodotto alcun output.

```
Remove-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

- Per i dettagli sull'API, vedere [DeleteAssociation](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-SSMDocument

Il seguente esempio di codice mostra come utilizzare. Remove-SSMDocument

Strumenti per PowerShell

Esempio 1: questo esempio elimina un documento. Non viene prodotto alcun risultato se il comando ha esito positivo.

```
Remove-SSMDocument -Name "RunShellScript"
```

- Per i dettagli sull'API, vedere [DeleteDocument](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-SSMMaintenanceWindow

Il seguente esempio di codice mostra come utilizzare. Remove-SSMMaintenanceWindow

Strumenti per PowerShell

Esempio 1: questo esempio rimuove una finestra di manutenzione.

```
Remove-SSMMaintenanceWindow -WindowId "mw-06d59c1a07c022145"
```

Output:

```
mw-06d59c1a07c022145
```

- Per i dettagli sull'API, vedere [DeleteMaintenanceWindow](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-SSMParameter

Il seguente esempio di codice mostra come utilizzare. Remove-SSMParameter

Strumenti per PowerShell

Esempio 1: questo esempio elimina un parametro. Se il comando ha esito positivo, non viene prodotto alcun output.

```
Remove-SSMParameter -Name "helloWorld"
```

- Per i dettagli sull'API, vedere [DeleteParameter](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-SSMPatchBaseline

Il seguente esempio di codice mostra come utilizzare. Remove-SSMPatchBaseline

Strumenti per PowerShell

Esempio 1: questo esempio elimina una patch baseline.

```
Remove-SSMPatchBaseline -BaselineId "pb-045f10b4f382baeda"
```

Output:

```
pb-045f10b4f382baeda
```

- Per i dettagli sull'API, vedere [DeletePatchBaseline](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-SSMResourceTag

Il seguente esempio di codice mostra come utilizzare. Remove-SSMResourceTag

Strumenti per PowerShell

Esempio 1: questo esempio rimuove un tag da una finestra di manutenzione. Se il comando ha esito positivo, non viene emesso alcun output.

```
Remove-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
  "MaintenanceWindow" -TagKey "Production"
```

- Per i dettagli sull'API, vedere [RemoveTagsFromResource](#) in AWS Tools for PowerShell Cmdlet Reference.

Send-SSMCommand

Il seguente esempio di codice mostra come utilizzare Send-SSMCommand

Strumenti per PowerShell

Esempio 1: questo esempio esegue un comando echo su un'istanza di destinazione.

```
Send-SSMCommand -DocumentName "AWS-RunPowerShellScript" -Parameter @{commands =
  "echo helloWorld"} -Target @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
```

Output:

```
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}
```

Esempio 2: questo esempio mostra come eseguire un comando che accetta parametri annidati.

```
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
```

- Per i dettagli sull'API, vedere [SendCommand](#) in AWS Tools for PowerShell Cmdlet Reference.

Start-SSMAutomationExecution

Il seguente esempio di codice mostra come utilizzare `Start-SSMAutomationExecution`

Strumenti per PowerShell

Esempio 1: questo esempio esegue un documento che specifica un ruolo di automazione, un ID di origine AMI e un ruolo di istanza Amazon EC2.

```
Start-SSMAutomationExecution -DocumentName AWS-UpdateLinuxAmi -
Parameter @{'AutomationAssumeRole'='arn:aws:iam::123456789012:role/
SSMAutomationRole';'SourceAmiId'='ami-f173cc91';'InstanceIamRole'='EC2InstanceRole'}
```

Output:

```
3a532a4f-0382-11e7-9df7-6f11185f6dd1
```

- Per i dettagli sull'API, vedere [StartAutomationExecution](#) in AWS Tools for PowerShell Cmdlet Reference.

Stop-SSMAutomationExecution

Il seguente esempio di codice mostra come utilizzare `Stop-SSMAutomationExecution`

Strumenti per PowerShell

Esempio 1: questo esempio interrompe un'esecuzione di automazione. Non viene prodotto alcun output se il comando ha esito positivo.

```
Stop-SSMAutomationExecution -AutomationExecutionId "4105a4fc-f944-11e6-9d32-8fb2db27a909"
```

- Per i dettagli sull'API, vedere [StopAutomationExecution](#) in AWS Tools for PowerShell Cmdlet Reference.

Stop-SSMCommand

Il seguente esempio di codice mostra come utilizzare. Stop-SSMCommand

Strumenti per PowerShell

Esempio 1: Questo esempio tenta di annullare un comando. Non viene emesso alcun risultato se l'operazione ha esito positivo.

```
Stop-SSMCommand -CommandId "9ded293e-e792-4440-8e3e-7b8ec5feaa38"
```

- Per i dettagli sull'API, vedere [CancelCommand](#) in AWS Tools for PowerShell Cmdlet Reference.

Unregister-SSManagedInstance

Il seguente esempio di codice mostra come utilizzare. Unregister-SSManagedInstance

Strumenti per PowerShell

Esempio 1: questo esempio annulla la registrazione di un'istanza gestita. Se il comando ha esito positivo, non viene prodotto alcun output.

```
Unregister-SSManagedInstance -InstanceId "mi-08ab247cdf1046573"
```

- Per i dettagli sull'API, vedere [DeregisterManagedInstance](#) in AWS Tools for PowerShell Cmdlet Reference.

Unregister-SSMPatchBaselineForPatchGroup

Il seguente esempio di codice mostra come utilizzare. Unregister-SSMPatchBaselineForPatchGroup

Strumenti per PowerShell

Esempio 1: Questo esempio annulla la registrazione di un gruppo di patch da una linea di base di patch.

```
Unregister-SSMPatchBaselineForPatchGroup -BaselineId "pb-045f10b4f382baeda" -
PatchGroup "Production"
```

Output:

```
BaselineId          PatchGroup
-----
pb-045f10b4f382baeda Production
```

- Per i dettagli sull'API, vedere [DeregisterPatchBaselineForPatchGroup](#) in Cmdlet Reference.AWS Tools for PowerShell

Unregister-SSMTargetFromMaintenanceWindow

Il seguente esempio di codice mostra come utilizzare. Unregister-SSMTargetFromMaintenanceWindow

Strumenti per PowerShell

Esempio 1: Questo esempio rimuove un obiettivo da una finestra di manutenzione.

```
Unregister-SSMTargetFromMaintenanceWindow -WindowTargetId "6ab5c208-9fc4-4697-84b7-
b02a6cc25f7d" -WindowId "mw-06cf17cbefcb4bf4f"
```

Output:

```
WindowId          WindowTargetId
-----
mw-06cf17cbefcb4bf4f 6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

- Per i dettagli sull'API, vedere [DeregisterTargetFromMaintenanceWindow](#) in AWS Tools for PowerShell Cmdlet Reference.

Unregister-SSMTaskFromMaintenanceWindow

Il seguente esempio di codice mostra come utilizzare. Unregister-SSMTaskFromMaintenanceWindow

Strumenti per PowerShell

Esempio 1: questo esempio rimuove un'attività da una finestra di manutenzione.

```
Unregister-SSMTaskFromMaintenanceWindow -WindowTaskId "f34a2c47-ddfd-4c85-a88d-72366b69af1b" -WindowId "mw-03a342e62c96d31b0"
```

Output:

```
WindowId           WindowTaskId
-----
mw-03a342e62c96d31b0 f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

- Per i dettagli sull'API, vedere [DeregisterTaskFromMaintenanceWindow](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-SSMAssociation

Il seguente esempio di codice mostra come utilizzare. Update-SSMAssociation

Strumenti per PowerShell

Esempio 1: questo esempio aggiorna un'associazione con una nuova versione del documento.

```
Update-SSMAssociation -AssociationId "93285663-92df-44cb-9f26-2292d4ecc439" -DocumentVersion "1"
```

Output:

```
Name           : AWS-UpdateSSMAgent
InstanceId      :
Date           : 3/1/2017 6:22:21 PM
Status.Name     :
Status.Date    :
Status.Message  :
Status.AdditionalInfo :
```


- Per i dettagli sull'API, vedere [UpdateAssociation](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-SSMAssociationStatus

Il seguente esempio di codice mostra come utilizzare. Update-SSMAssociationStatus

Strumenti per PowerShell

Esempio 1: Questo esempio aggiorna lo stato dell'associazione tra un'istanza e un documento di configurazione.

```
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"
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [UpdateAssociationStatus](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-SSMDocument

Il seguente esempio di codice mostra come utilizzare. Update-SSMDocument

Strumenti per PowerShell

Esempio 1: viene creata una nuova versione di un documento con il contenuto aggiornato del file json specificato. Il documento deve essere in formato JSON. È possibile ottenere la versione del documento con il cmdlet «Get-SSMDocumentVersionList».

```
Update-SSMDocument -Name RunShellScript -DocumentVersion "1" -Content (Get-Content -Raw "c:\temp\RunShellScript.json")
```

Output:

```

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

```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [UpdateDocument](#) AWS Tools for PowerShell

Update-SSMDocumentDefaultVersion

Il seguente esempio di codice mostra come utilizzare. `Update-SSMDocumentDefaultVersion`

Strumenti per PowerShell

Esempio 1: Questo aggiorna la versione predefinita di un documento. È possibile ottenere le versioni dei documenti disponibili con il cmdlet «`Get-SSMDocumentVersionList`».

```
Update-SSMDocumentDefaultVersion -Name "RunShellScript" -DocumentVersion "2"
```

Output:

```

DefaultVersion Name
-----
2              RunShellScript

```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [UpdateDocumentDefaultVersion](#) AWS Tools for PowerShell

Update-SSMMaintenanceWindow

Il seguente esempio di codice mostra come utilizzare. Update-SSMMaintenanceWindow

Strumenti per PowerShell

Esempio 1: questo esempio aggiorna il nome di una finestra di manutenzione.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Name "My-Renamed-MW"
```

Output:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

Esempio 2: questo esempio abilita una finestra di manutenzione.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $true
```

Output:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

Esempio 3: questo esempio disabilita una finestra di manutenzione.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $false
```

Output:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : False
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

- Per i dettagli sull'API, vedere [UpdateMaintenanceWindow](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-SSMManagedInstanceRole

Il seguente esempio di codice mostra come utilizzare `Update-SSMManagedInstanceRole`

Strumenti per PowerShell

Esempio 1: questo esempio aggiorna il ruolo di un'istanza gestita. Se il comando ha esito positivo, non viene prodotto alcun output.

```
Update-SSMManagedInstanceRole -InstanceId "mi-08ab247cdf1046573" -IamRole
"AutomationRole"
```

- Per i dettagli sull'API, vedere [UpdateManagedInstanceRole](#) in AWS Tools for PowerShell Cmdlet Reference.

Update-SSMPatchBaseline

Il seguente esempio di codice mostra come utilizzare `Update-SSMPatchBaseline`

Strumenti per PowerShell

Esempio 1: Questo esempio aggiunge due patch come rifiutate e una patch approvata a una patch di base esistente.

```
Update-SSMPatchBaseline -BaselineId "pb-03da896ca3b68b639" -RejectedPatch
"KB2032276", "MS10-048" -ApprovedPatch "KB2124261"
```

Output:

```
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}
```

- Per i dettagli sull'API, vedere [UpdatePatchBaseline](#) in AWS Tools for PowerShell Cmdlet Reference.

Write-SSMComplianceItem

Il seguente esempio di codice mostra come utilizzare `Write-SSMComplianceItem`

Strumenti per PowerShell

Esempio 1: questo esempio scrive un elemento di conformità personalizzato per l'istanza gestita specificata

```
$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"
```

- Per i dettagli sull'API, vedere [PutComplianceItems](#) in AWS Tools for PowerShell Cmdlet Reference.

Write-SSMInventory

Il seguente esempio di codice mostra come utilizzare `Write-SSMInventory`

Strumenti per PowerShell

Esempio 1: questo esempio assegna le informazioni sulla posizione del rack a un'istanza. Se il comando ha esito positivo, non viene prodotto alcun output.

```
$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
```

- Per i dettagli sull'API, vedere [PutInventory](#) in AWS Tools for PowerShell Cmdlet Reference.

Write-SSMParameter

Il seguente esempio di codice mostra come utilizzare `Write-SSMParameter`

Strumenti per PowerShell

Esempio 1: Questo esempio crea un parametro. Se il comando ha esito positivo, non viene prodotto alcun output.

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "helloWorld"
```

Esempio 2: questo esempio modifica un parametro. Non viene emesso alcun risultato se il comando ha esito positivo.

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "Good day, Sunshine!" -  
Overwrite $true
```

- Per i dettagli sull'API, vedere [PutParameter](#) in AWS Tools for PowerShell Cmdlet Reference.

Esempi di Amazon Translate con Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon Translate. AWS Tools for PowerShell

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

ConvertTo-TRNTargetLanguage

Il seguente esempio di codice mostra come utilizzare `ConvertTo-TRNTargetLanguage`.

Strumenti per PowerShell

Esempio 1: converte il testo inglese specificato in francese. Il testo da convertire può anche essere passato come parametro `-Text`.

```
"Hello World" | ConvertTo-TRNTargetLanguage -SourceLanguageCode en -  
TargetLanguageCode fr
```

- Per i dettagli sull'API, vedere [TranslateText](#) in AWS Tools for PowerShell Cmdlet Reference.

AWS WAFV2 esempi di utilizzo di Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with AWS WAFV2.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

New-WAF2WebACL

Il seguente esempio di codice mostra come utilizzare New-WAF2WebACL.

Strumenti per PowerShell

Esempio 1: questo comando crea un nuovo ACL web denominato «waf-test». Tieni presente che, secondo la documentazione dell'API di servizio, 'DefaultAction' è una proprietà obbligatoria. Pertanto, è necessario specificare il valore per '- DefaultAction _Allow' e/o '- DefaultAction _Block'. Poiché '- DefaultAction _Allow' e '- DefaultAction _Block' non sono le proprietà richieste, il valore '@ {}' potrebbe essere usato come segnaposto come mostrato nell'esempio precedente.

```
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 @{}
```

Output:

```
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
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [CreateWebAcl](#) AWS Tools for PowerShell

WorkSpaces esempi di utilizzo di Tools for PowerShell

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS Tools for PowerShell with WorkSpaces.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

Approve-WKSIpRule

Il seguente esempio di codice mostra come utilizzare `Approve-WKSIpRule`.

Strumenti per PowerShell

Esempio 1: questo esempio aggiunge regole a un gruppo IP esistente

```
$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
```

- Per i dettagli sull'API, vedere [AuthorizeIpRules](#) in AWS Tools for PowerShell Cmdlet Reference.

Copy-WKSWorkspaceImage

Il seguente esempio di codice mostra come utilizzare. Copy-WKSWorkspaceImage

Strumenti per PowerShell

Esempio 1: questo esempio copia l'immagine dell'area di lavoro con l'ID specificato da us-west-2 nella regione corrente con il nome "" CopiedImageTest

```
Copy-WKSWorkspaceImage -Name CopiedImageTest -SourceRegion us-west-2 -SourceImageId wsi-djfoedhw6
```

Output:

```
wsi-456abaqfe
```

- Per i dettagli sull'API, vedere [CopyWorkspaceImage](#) in AWS Tools for PowerShell Cmdlet Reference.

Edit-WKSClientProperty

Il seguente esempio di codice mostra come utilizzare. Edit-WKSClientProperty

Strumenti per PowerShell

Esempio 1: questo esempio abilita la riconnessione per il client Workspaces

```
Edit-WKSClientProperty -Region us-west-2 -ClientProperties_ReconnectEnabled "ENABLED" -ResourceId d-123414a369
```

- Per i dettagli sull'API, vedere [ModifyClientProperties](#) in AWS Tools for PowerShell Cmdlet Reference.

Edit-WKSSelfServicePermission

Il seguente esempio di codice mostra come utilizzare. Edit-WKSSelfServicePermission

Strumenti per PowerShell

Esempio 1: Questo esempio abilita le autorizzazioni self-service per modificare il tipo di elaborazione e aumentare le dimensioni del volume per la directory specificata

```
Edit-WKSSelfservicePermission -Region us-west-2 -ResourceId  
d-123454a369 -SelfservicePermissions_ChangeComputeType ENABLED -  
SelfservicePermissions_IncreaseVolumeSize ENABLED
```

- Per i dettagli sull'API, vedere [ModifySelfservicePermissions](#) in AWS Tools for PowerShell Cmdlet Reference.

Edit-WKSWorkspaceAccessProperty

Il seguente esempio di codice mostra come utilizzare. Edit-WKSWorkspaceAccessProperty

Strumenti per PowerShell

Esempio 1: questo esempio abilita l'accesso all'area di lavoro su Android e Chrome OS per la directory specificata

```
Edit-WKSWorkspaceAccessProperty -Region us-west-2 -ResourceId  
d-123454a369 -WorkspaceAccessProperties_DeviceTypeAndroid ALLOW -  
WorkspaceAccessProperties_DeviceTypeChromeOs ALLOW
```

- Per i dettagli sull'API, vedere [ModifyWorkspaceAccessProperties](#) in AWS Tools for PowerShell Cmdlet Reference.

Edit-WKSWorkspaceCreationProperty

Il seguente esempio di codice mostra come utilizzare. Edit-WKSWorkspaceCreationProperty

Strumenti per PowerShell

Esempio 1: questo esempio abilita l'accesso a Internet e la modalità di manutenzione su true come valori predefiniti durante la creazione di un'area di lavoro

```
Edit-WKSWorkspaceCreationProperty -Region us-west-2 -ResourceId  
d-123454a369 -WorkspaceCreationProperties_EnableInternetAccess $true -  
WorkspaceCreationProperties_EnableMaintenanceMode $true
```

- Per i dettagli sull'API, vedere [ModifyWorkspaceCreationProperties](#) in AWS Tools for PowerShell Cmdlet Reference.

Edit-WKSWorkspaceProperty

Il seguente esempio di codice mostra come utilizzare. Edit-WKSWorkspaceProperty

Strumenti per PowerShell

Esempio 1: questo esempio modifica la proprietà Workspace Running Mode in Auto Stop per l'area di lavoro specificata

```
Edit-WKSWorkspaceProperty -WorkspaceId ws-w361s100v -Region us-west-2 -  
WorkspaceProperties_RunningMode AUTO_STOP
```

- Per i dettagli sull'API, vedere [ModifyWorkspaceProperties](#) in AWS Tools for PowerShell Cmdlet Reference.

Edit-WKSWorkspaceState

Il seguente esempio di codice mostra come utilizzare. Edit-WKSWorkspaceState

Strumenti per PowerShell

Esempio 1: questo esempio modifica lo stato dell'area di lavoro specificata in Disponibile

```
Edit-WKSWorkspaceState -WorkspaceId ws-w361s100v -Region us-west-2 -WorkspaceState  
AVAILABLE
```

- Per i dettagli sull'API, vedere [ModifyWorkspaceState](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-WKSClientProperty

Il seguente esempio di codice mostra come utilizzare. Get-WKSClientProperty

Strumenti per PowerShell

Esempio 1: Questo esempio ottiene le proprietà del client Workspace per la directory specificata

```
Get-WKSCClientProperty -ResourceId d-223562a123
```

- Per i dettagli sull'API, vedere [DescribeClientProperties](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-WKSIpGroup

Il seguente esempio di codice mostra come utilizzare. Get-WKSIpGroup

Strumenti per PowerShell

Esempio 1: questo esempio ottiene i dettagli del gruppo IP specificato nella regione specificata

```
Get-WKSIpGroup -Region us-east-1 -GroupId wsipg-8m1234v45
```

Output:

```
GroupDesc GroupId      GroupName UserRules
-----
wsipg-8m1234v45 TestGroup {Amazon.WorkSpaces.Model.IpRuleItem,
Amazon.WorkSpaces.Model.IpRuleItem}
```

- Per i dettagli sull'API, vedere [DescribeIpGroups](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-WKSTag

Il seguente esempio di codice mostra come utilizzare. Get-WKSTag

Strumenti per PowerShell

Esempio 1: questo esempio recupera il tag per l'area di lavoro specificata

```
Get-WKSTag -WorkspaceId ws-w361s234r -Region us-west-2
```

Output:

```
Key      Value
---      -
```

```
auto-delete no
purpose      Workbench
```

- Per i dettagli sull'API, vedere [DescribeTags](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-WKSWorkspace

Il seguente esempio di codice mostra come utilizzare. Get-WKSWorkspace

Strumenti per PowerShell

Esempio 1: recupera i dettagli di tutti gli accessi WorkSpaces alla pipeline.

```
Get-WKSWorkspace
```

Output:

```
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
```

Esempio 2: Questo comando mostra i valori delle proprietà secondarie di un'area **WorkspaceProperties** di lavoro nella regione. **us-west-2** Per ulteriori informazioni sulle proprietà secondarie di **WorkspaceProperties**, vedete https://docs.aws.amazon.com/workspaces/latest/api/API_WorkspaceProperties.html.

```
(Get-WKSWorkspace -Region us-west-2 -WorkspaceId ws-xdaf7hc9s).WorkspaceProperties
```

Output:

```
ComputeTypeName           : STANDARD
RootVolumeSizeGib        : 80
RunningMode                : AUTO_STOP
RunningModeAutoStopTimeoutInMinutes : 60
UserVolumeSizeGib        : 50
```

Esempio 3: questo comando mostra il valore della proprietà child **RootVolumeSizeGib** di **WorkspaceProperties** per un'area di lavoro nella **us-west-2** regione. La dimensione del volume root, in GiB, è 80.

```
(Get-WKSWorkspace -Region us-west-2 -WorkspaceId ws-
xdaf7hc9s).WorkspaceProperties.RootVolumeSizeGib
```

Output:

```
80
```

- Per i dettagli sull'API, vedere [DescribeWorkspaces](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-WKSWorkspaceBundle

Il seguente esempio di codice mostra come utilizzare `Get-WKSWorkspaceBundle`

Strumenti per PowerShell

Esempio 1: questo esempio recupera i dettagli di tutti i pacchetti Workspace nell'area corrente

```
Get-WKSWorkspaceBundle
```

Output:

```
BundleId           : wsb-sfhgfv342
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
```

- Per i dettagli sull'API, vedere [DescribeWorkspaceBundles](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-WKSWorkspaceDirectory

Il seguente esempio di codice mostra come utilizzare. `Get-WKSWorkspaceDirectory`

Strumenti per PowerShell

Esempio 1: questo esempio elenca i dettagli delle directory registrate

```
Get-WKSWorkspaceDirectory
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DescribeWorkspaceDirectories](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-WKSWorkspaceImage

Il seguente esempio di codice mostra come utilizzare. `Get-WKSWorkspaceImage`

Strumenti per PowerShell

Esempio 1: questo esempio recupera tutti i dettagli di tutte le immagini della regione

```
Get-WKSWorkspaceImage
```

Output:

```
Description      :This image is copied from another image
ErrorCode        :
ErrorMessage     :
ImageId          : wsi-345ahdjgo
Name             : CopiedImageTest
OperatingSystem  : Amazon.WorkSpaces.Model.OperatingSystem
RequiredTenancy  : DEFAULT
State            : AVAILABLE
```

- Per i dettagli sull'API, vedere [DescribeWorkspaceImages](#) in AWS Tools for PowerShell Cmdlet Reference.

Get-WKSWorkspaceSnapshot

Il seguente esempio di codice mostra come utilizzare. Get-WKSWorkspaceSnapshot

Strumenti per PowerShell

Esempio 1: Questo esempio mostra il timestamp dell'istantanea più recente creata per l'area di lavoro specificata

```
Get-WKSWorkspaceSnapshot -WorkspaceId ws-w361s100v
```

Output:

```
RebuildSnapshots          RestoreSnapshots
-----
{Amazon.WorkSpaces.Model.Snapshot} {Amazon.WorkSpaces.Model.Snapshot}
```

- Per i dettagli sull'API, vedere [DescribeWorkspaceSnapshots](#) in Cmdlet Reference.AWS Tools for PowerShell

Get-WKSWorkspacesConnectionStatus

Il seguente esempio di codice mostra come utilizzare. `Get-WKSWorkspacesConnectionStatus`

Strumenti per PowerShell

Esempio 1: questo esempio recupera lo stato della connessione per l'area di lavoro specificata

```
Get-WKSWorkspacesConnectionStatus -WorkspaceId ws-w123s234r
```

- Per i dettagli sull'API, vedere [DescribeWorkspacesConnectionStatus](#) in AWS Tools for PowerShell Cmdlet Reference.

New-WKSIpGroup

Il seguente esempio di codice mostra come utilizzare. `New-WKSIpGroup`

Strumenti per PowerShell

Esempio 1: questo esempio crea un gruppo Ip vuoto denominato `FreshEmptyIpGroup`

```
New-WKSIpGroup -GroupName "FreshNewIPGroup"
```

Output:

```
wsipg-w45rty4ty
```

- Per i dettagli sull'API, vedere [CreateIpGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

New-WKSTag

Il seguente esempio di codice mostra come utilizzare. `New-WKSTag`

Strumenti per PowerShell

Esempio 1: Questo esempio aggiunge un nuovo tag a un'area di lavoro denominata `aws-wsname`. Il tag ha una chiave di «Nome» e un valore chiave di `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
```

Esempio 2: Questo esempio aggiunge più tag a un'area di lavoro denominata **aws-wsname**. Un tag ha una chiave «Name» e un valore chiave di **AWS_Workspace**; l'altro tag ha una chiave di tag «Stage» e un valore chiave «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
```

- Per i dettagli sull'API, vedere [CreateTags](#) in AWS Tools for PowerShell Cmdlet Reference.

New-WKSWorkspace

Il seguente esempio di codice mostra come utilizzare `New-WKSWorkspace`

Strumenti per PowerShell

Esempio 1: crea un file WorkSpace per il pacchetto, la directory e l'utente forniti.

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId" =
"d-1a2b3c4d"; "UserName" = "USERNAME"}
```

Esempio 2: Questo esempio crea più WorkSpaces

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId"
= "d-1a2b3c4d"; "UserName" = "USERNAME_1"},@{"BundleID" = "wsb-1a2b3c4d";
"DirectoryId" = "d-1a2b3c4d"; "UserName" = "USERNAME_2"}
```

- Per i dettagli sull'API, vedere [CreateWorkspaces](#) in AWS Tools for PowerShell Cmdlet Reference.

Register-WKSipGroup

Il seguente esempio di codice mostra come utilizzare `Register-WKSipGroup`

Strumenti per PowerShell

Esempio 1: questo esempio registra il gruppo IP specificato nella directory specificata

```
Register-WKSIpGroup -GroupId wsipg-23ahsdres -DirectoryId d-123412e123
```

- Per i dettagli sull'API, vedere [AssociateIpGroups](#) in AWS Tools for PowerShell Cmdlet Reference.

Register-WKSWorkspaceDirectory

Il seguente esempio di codice mostra come utilizzare Register-WKSWorkspaceDirectory

Strumenti per PowerShell

Esempio 1: questo esempio registra la directory specificata per il servizio Workspaces

```
Register-WKSWorkspaceDirectory -DirectoryId d-123412a123 -EnableWorkDoc $false
```

- Per i dettagli sull'API, vedere [RegisterWorkspaceDirectory](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-WKSIpGroup

Il seguente esempio di codice mostra come utilizzare Remove-WKSIpGroup

Strumenti per PowerShell

Esempio 1: questo esempio elimina il gruppo IP specificato

```
Remove-WKSIpGroup -GroupId wsipg-32fhgtred
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DeleteIpGroup](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-WKSTag

Il seguente esempio di codice mostra come utilizzare. Remove-WKSTag

Strumenti per PowerShell

Esempio 1: questo esempio rimuove il tag associato all'area di lavoro

```
Remove-WKSTag -ResourceId ws-w10b3abcd -TagKey "Type"
```

Output:

```
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
```

- Per i dettagli sull'API, vedere [DeleteTags](#) in AWS Tools for PowerShell Cmdlet Reference.

Remove-WKSWorkspace

Il seguente esempio di codice mostra come utilizzare. Remove-WKSWorkspace

Strumenti per PowerShell

Esempio 1: termina più elementi WorkSpaces. L'utilizzo dell'opzione -Force impedisce al cmdlet di richiedere la conferma.

```
Remove-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5", "ws-6a7b8c9d0" -Force
```

Esempio 2: recupera la raccolta di tutti i tuoi WorkSpaces e reindirizza gli ID al WorkspaceId parametro - di Remove-WKSWorkspace, terminando tutti i WorkSpaces Il cmdlet verrà richiesto prima che ciascuna di esse venga terminata. Workspace Per sopprimere la richiesta di conferma, aggiungere l'opzione -Force.

```
Get-WKSWorkspaces | Remove-WKSWorkspace
```

Esempio 3: Questo esempio mostra come passare `TerminateRequest` oggetti che definiscono l'oggetto `WorkSpaces` da terminare. Il cmdlet richiederà una conferma prima di procedere, a meno che non venga specificato anche il parametro `-Force` switch.

```
$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
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [TerminateWorkspaces](#) AWS Tools for PowerShell

Reset-WKSWorkspace

Il seguente esempio di codice mostra come utilizzare. `Reset-WKSWorkspace`

Strumenti per PowerShell

Esempio 1: ricostruisce ciò che è specificato. `Workspace`

```
Reset-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

Esempio 2: recupera la raccolta di tutti i tuoi dati `WorkSpaces` e reindirizza gli ID al `WorkspaceId` parametro - di `Reset-WKSWorkspace`, provocandone la ricostruzione. `WorkSpaces`

```
Get-WKSWorkspaces | Reset-WKSWorkspace
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [RebuildWorkspaces](#) AWS Tools for PowerShell

Restart-WKSWorkspace

Il seguente esempio di codice mostra come utilizzare. `Restart-WKSWorkspace`

Strumenti per PowerShell

Esempio 1: riavvia il file specificato Workspace.

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

Esempio 2: riavvia più volte. WorkSpaces

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d","ws-5a6b7c8d"
```

Esempio 3: recupera la raccolta di tutti i file WorkSpaces e reindirizza gli ID al WorkspaceId parametro - di Restart-WKSWorkspace, provocandone il riavvio. WorkSpaces

```
Get-WKSWorkspaces | Restart-WKSWorkspace
```

- Per i dettagli sull'API, vedere in Cmdlet Reference. [RebootWorkspaces](#) AWS Tools for PowerShell

Stop-WKSWorkspace

Il seguente esempio di codice mostra come utilizzare. Stop-WKSWorkspace

Strumenti per PowerShell

Esempio 1: interrompe più interruzioni WorkSpaces.

```
Stop-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0"
```

Esempio 2: recupera la raccolta di tutti i tuoi dati WorkSpaces e reindirizza gli ID al WorkspaceId parametro - di Stop-WKSWorkspace che causa l'interruzione. WorkSpaces

```
Get-WKSWorkspaces | Stop-WKSWorkspace
```

Esempio 3: Questo esempio mostra come passare StopRequest oggetti che definiscono il punto da fermare. 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
```

- Per i dettagli sull'API, vedere [StopWorkspaces](#) in AWS Tools for PowerShell Cmdlet Reference.

Unregister-WKSIpGroup

Il seguente esempio di codice mostra come utilizzare `Unregister-WKSIpGroup`

Strumenti per PowerShell

Esempio 1: questo esempio annulla la registrazione del gruppo IP specificato dalla directory specificata

```
Unregister-WKSIpGroup -GroupId wsipg-12abcdphq -DirectoryId d-123454b123
```

- Per i dettagli sull'API, vedere [DisassociateIpGroups](#) in AWS Tools for PowerShell Cmdlet Reference.

Sicurezza per questo AWS prodotto o servizio

La sicurezza cloud di Amazon Web Services (AWS) è la priorità più alta. In quanto cliente AWS, è possibile trarre vantaggio da un'architettura di data center e di rete progettata per soddisfare i requisiti delle organizzazioni più esigenti a livello di sicurezza. La sicurezza è una responsabilità condivisa tra AWS e te. Il [modello di responsabilità condivisa](#) descrive questo come sicurezza del cloud e sicurezza nel cloud.

Security of the Cloud: AWS è responsabile della protezione dell'infrastruttura che gestisce tutti i servizi offerti nel AWS Cloud e della fornitura di servizi che è possibile utilizzare in modo sicuro. La nostra responsabilità in AWS materia di sicurezza è la massima priorità e l'efficacia della nostra sicurezza viene regolarmente testata e verificata da revisori di terze parti nell'ambito dei Programmi di [AWS conformità](#).

Sicurezza nel cloud: la responsabilità dell'utente è determinata dal AWS servizio utilizzato e da altri fattori, tra cui la sensibilità dei dati, i requisiti dell'organizzazione e le leggi e i regolamenti applicabili.

Questo AWS prodotto o servizio segue il [modello di responsabilità condivisa](#) attraverso i servizi specifici di Amazon Web Services (AWS) che supporta. Per informazioni sulla sicurezza dei AWS servizi, consulta la [pagina della documentazione sulla sicurezza del AWS servizio](#) e [AWS i servizi che rientrano nell'ambito delle iniziative di AWS conformità previste dal programma di conformità](#).

Argomenti

- [Protezione dei dati in questo prodotto o servizio AWS](#)
- [Identity and Access Management](#)
- [Convalida della conformità per questo prodotto o servizio AWS](#)
- [Applicazione di una versione minima di TLS in Strumenti per PowerShell](#)
- [Considerazioni aggiuntive sulla sicurezza per gli strumenti per PowerShell](#)

Protezione dei dati in questo prodotto o servizio AWS

Il [modello di responsabilità condivisa](#) AWS si applica alla protezione dei dati in questo prodotto o servizio AWS. Come descritto in questo modello, AWS è responsabile della protezione dell'infrastruttura globale che esegue tutto l'Cloud AWS. L'utente è responsabile del controllo dei contenuti ospitati su questa infrastruttura. Inoltre, sei responsabile della configurazione della protezione e delle attività di gestione per i Servizi AWS che utilizzi. Per ulteriori informazioni sulla

privacy dei dati, vedi le [Domande frequenti sulla privacy dei dati](#). Per informazioni sulla protezione dei dati in Europa, consulta il post del blog relativo al [Modello di responsabilità condivisa AWS e GDPR](#) nel Blog sulla sicurezza AWS.

Per garantire la protezione dei dati, ti suggeriamo di proteggere le credenziali Account AWS e di configurare singoli utenti con AWS IAM Identity Center o AWS Identity and Access Management (IAM). In tal modo, a ogni utente verranno assegnate solo le autorizzazioni necessarie per svolgere i suoi compiti. Ti suggeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Utilizza l'autenticazione a più fattori (MFA) con ogni account.
- Utilizza SSL/TLS per comunicare con le risorse AWS. È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Configura l'API e la registrazione delle attività degli utenti con AWS CloudTrail.
- Utilizza le soluzioni di crittografia AWS, insieme a tutti i controlli di sicurezza predefiniti in Servizi AWS.
- Utilizza i servizi di sicurezza gestiti avanzati, come Amazon Macie, che aiutano a individuare e proteggere i dati sensibili archiviati in Amazon S3.
- Se necessiti di moduli crittografici convalidati FIPS 140-2 quando accedi ad AWS attraverso un'interfaccia a riga di comando o un'API, utilizza un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il [Federal Information Processing Standard \(FIPS\) 140-2](#).

Ti consigliamo vivamente di non inserire mai informazioni riservate o sensibili, ad esempio gli indirizzi e-mail dei clienti, nei tag o nei campi di testo in formato libero, ad esempio nel campo Nome. Ciò include quando utilizzi questo prodotto o servizio AWS o altri Servizi AWS mediante la console, l'API, la AWS CLI o gli SDK AWS. I dati inseriti nei tag o nei campi di testo in formato libero utilizzati per i nomi possono essere utilizzati per la fatturazione o i log di diagnostica. Quando fornisci un URL a un server esterno, ti suggeriamo vivamente di non includere informazioni sulle credenziali nell'URL per convalidare la tua richiesta al server.

Crittografia dei dati

Una caratteristica fondamentale di qualsiasi servizio sicuro è che le informazioni vengano crittografate quando non sono utilizzate attivamente.

Crittografia dei dati inattivi

AWS Tools for PowerShell non memorizza dati del cliente diversi dalle credenziali di cui ha bisogno per interagire con i servizi AWS per conto dell'utente.

Se utilizzi AWS Tools for PowerShell per richiamare un servizio AWS che trasmette i dati dei clienti al computer locale per l'archiviazione, fai riferimento al capitolo “Sicurezza e conformità” della Guida per l'utente di tale servizio per informazioni su come tali dati vengono archiviati, protetti e crittografati.

Crittografia in transito

Per impostazione predefinita, tutti i dati trasmessi dal computer client che esegue gli endpoint di AWS Tools for PowerShell e del servizio AWS vengono crittografati inviando ogni elemento tramite una connessione HTTPS/TLS.

Non devi fare nulla per abilitare l'uso di HTTPS/TLS. È sempre abilitato.

Identity and Access Management

AWS Identity and Access Management (IAM) è un Servizio AWS che consente agli amministratori di controllare in modo sicuro l'accesso alle risorse AWS. Gli amministratori IAM controllano chi è autenticato (accesso effettuato) e autorizzato (dispone di autorizzazioni) a utilizzare risorse AWS. IAM è un Servizio AWS che è possibile utilizzare senza alcun costo aggiuntivo.

Argomenti

- [Destinatari](#)
- [Autenticazione con identità](#)
- [Gestione dell'accesso con policy](#)
- [Funzionamento dei Servizi AWS con IAM](#)
- [Risoluzione dei problemi di identità e accesso in AWS](#)

Destinatari

Le modalità di utilizzo di AWS Identity and Access Management (IAM) cambiano in base alle operazioni eseguite in AWS.

Utente del servizio: se utilizzi i Servizi AWS per eseguire il tuo lavoro, l'amministratore fornisce le credenziali e le autorizzazioni necessarie. All'aumentare del numero di funzionalità AWS utilizzate per il lavoro, potrebbero essere necessarie ulteriori autorizzazioni. La comprensione della gestione dell'accesso ti consente di richiedere le autorizzazioni corrette all'amministratore. Se non riesci ad accedere a una funzionalità in AWS, consulta la pagina [Risoluzione dei problemi di identità e accesso in AWS](#) oppure la guida per l'utente del Servizio AWS che stai utilizzando.

Amministratore del servizio: se sei il responsabile delle risorse AWS presso la tua azienda, probabilmente disponi dell'accesso completo a AWS. Il tuo compito è determinare le caratteristiche e le risorse AWS a cui gli utenti del servizio devono accedere. Devi inviare le richieste all'amministratore IAM per cambiare le autorizzazioni degli utenti del servizio. Esamina le informazioni contenute in questa pagina per comprendere i concetti di base relativi a IAM. Per ulteriori informazioni su come la tua azienda può utilizzare IAM con AWS, consulta la guida per l'utente del Servizio AWS che stai utilizzando.

Amministratore IAM: un amministratore IAM potrebbe essere interessato a ottenere dei dettagli su come scrivere policy per gestire l'accesso a AWS. Per visualizzare esempi di policy basate sull'identità di AWS che puoi utilizzare in IAM, consulta la guida per l'utente del Servizio AWS che stai utilizzando.

Autenticazione con identità

L'autenticazione è la procedura di accesso ad AWS con le credenziali di identità. Devi essere autenticato (connesso a AWS) come utente root Utente root dell'account AWS, come utente IAM o assumere un ruolo IAM.

Puoi accedere ad AWS come identità federata utilizzando le credenziali fornite attraverso un'origine di identità. Gli utenti AWS IAM Identity Center (Centro identità IAM), l'autenticazione Single Sign-On (SSO) dell'azienda e le credenziali di Google o Facebook sono esempi di identità federate. Se accedi come identità federata, l'amministratore ha configurato in precedenza la federazione delle identità utilizzando i ruoli IAM. Se accedi ad AWS tramite la federazione, assumi indirettamente un ruolo.

A seconda del tipo di utente, puoi accedere alla AWS Management Console o al portale di accesso AWS. Per ulteriori informazioni sull'accesso ad AWS, consulta la sezione [Come accedere al tuo Account AWS](#) nella Guida per l'utente di Accedi ad AWS.

Se accedi ad AWS in modo programmatico, AWS fornisce un Software Development Kit (SDK) e un'interfaccia a riga di comando (CLI) per firmare crittograficamente le richieste utilizzando le tue credenziali. Se non utilizzi gli strumenti AWS, devi firmare le richieste personalmente. Per ulteriori informazioni sulla firma delle richieste, consulta [Firma delle richieste AWS](#) nella Guida per l'utente IAM.

A prescindere dal metodo di autenticazione utilizzato, potrebbe essere necessario specificare ulteriori informazioni sulla sicurezza. AWS consiglia ad esempio di utilizzare l'autenticazione a più fattori (MFA) per aumentare la sicurezza dell'account. Per ulteriori informazioni, consulta [Autenticazione](#)

[a più fattori](#) nella Guida per l'utente di AWS IAM Identity Center e [Utilizzo dell'autenticazione a più fattori \(MFA\) in AWS](#) nella Guida per l'utente di IAM.

Utente root di un Account AWS

Quando crei un Account AWS, inizi con una singola identità di accesso che ha accesso completo a tutti i Servizi AWS e le risorse nell'account. Tale identità è detta utente root Account AWS ed è possibile accedervi con l'indirizzo e-mail e la password utilizzati per creare l'account. Si consiglia vivamente di non utilizzare l'utente root per le attività quotidiane. Conserva le credenziali dell'utente root e utilizzarle per eseguire le operazioni che solo l'utente root può eseguire. Per un elenco completo delle attività che richiedono l'accesso come utente root, consulta la sezione [Attività che richiedono le credenziali dell'utente root](#) nella Guida per l'utente di IAM.

Identità federata

Come best practice, richiedi agli utenti umani, compresi quelli che richiedono l'accesso di amministratore, di utilizzare la federazione con un provider di identità per accedere a Servizi AWS utilizzando credenziali temporanee.

Un'identità federata è un utente della directory degli utenti aziendali, un provider di identità Web, AWS Directory Service, la directory Identity Center o qualsiasi utente che accede ai Servizi AWS utilizzando le credenziali fornite tramite un'origine di identità. Quando le identità federate accedono agli Account AWS, assumono ruoli e i ruoli forniscono credenziali temporanee.

Per la gestione centralizzata degli accessi, consigliamo di utilizzare AWS IAM Identity Center. È possibile creare utenti e gruppi in IAM Identity Center oppure connettersi e sincronizzarsi con un gruppo di utenti e gruppi nell'origine di identità per utilizzarli in tutte le applicazioni e gli Account AWS. Per ulteriori informazioni sul Centro identità IAM, consulta [Cos'è Centro identità IAM?](#) nella Guida per l'utente di AWS IAM Identity Center.

Utenti e gruppi IAM

Un [utente IAM](#) è una identità all'interno del tuo Account AWS che dispone di autorizzazioni specifiche per una singola persona o applicazione. Ove possibile, consigliamo di fare affidamento a credenziali temporanee invece di creare utenti IAM con credenziali a lungo termine come le password e le chiavi di accesso. Tuttavia, per casi d'uso specifici che richiedono credenziali a lungo termine con utenti IAM, si consiglia di ruotare le chiavi di accesso. Per ulteriori informazioni, consulta la pagina [Rotazione periodica delle chiavi di accesso per casi d'uso che richiedono credenziali a lungo termine](#) nella Guida per l'utente di IAM.

Un [gruppo IAM](#) è un'identità che specifica un insieme di utenti IAM. Non è possibile eseguire l'accesso come gruppo. È possibile utilizzare gruppi per specificare le autorizzazioni per più utenti alla volta. I gruppi semplificano la gestione delle autorizzazioni per set di utenti di grandi dimensioni. Ad esempio, è possibile avere un gruppo denominato Amministratori IAM e concedere a tale gruppo le autorizzazioni per amministrare le risorse IAM.

Gli utenti sono diversi dai ruoli. Un utente è associato in modo univoco a una persona o un'applicazione, mentre un ruolo è destinato a essere assunto da chiunque ne abbia bisogno. Gli utenti dispongono di credenziali a lungo termine permanenti, mentre i ruoli forniscono credenziali temporanee. Per ulteriori informazioni, consulta [Quando creare un utente IAM \(invece di un ruolo\)](#) nella Guida per l'utente di IAM.

Ruoli IAM

Un [ruolo IAM](#) è un'identità all'interno di un Account AWS che dispone di autorizzazioni specifiche. È simile a un utente IAM, ma non è associato a una persona specifica. È possibile assumere temporaneamente un ruolo IAM nella AWS Management Console mediante lo [scambio di ruoli](#). È possibile assumere un ruolo chiamando un'azione AWS CLI o API AWS oppure utilizzando un URL personalizzato. Per ulteriori informazioni sui metodi per l'utilizzo dei ruoli, consulta [Utilizzo di ruoli IAM](#) nella Guida per l'utente di IAM.

I ruoli IAM con credenziali temporanee sono utili nelle seguenti situazioni:

- **Accesso utente federato:** per assegnare le autorizzazioni a una identità federata, è possibile creare un ruolo e definire le autorizzazioni per il ruolo. Quando un'identità federata viene autenticata, l'identità viene associata al ruolo e ottiene le autorizzazioni da esso definite. Per ulteriori informazioni sulla federazione dei ruoli, consulta [Creazione di un ruolo per un provider di identità di terza parte](#) nella Guida per l'utente di IAM. Se utilizzi IAM Identity Center, configura un set di autorizzazioni. IAM Identity Center mette in correlazione il set di autorizzazioni con un ruolo in IAM per controllare a cosa possono accedere le identità dopo l'autenticazione. Per ulteriori informazioni sui set di autorizzazioni, consulta [Set di autorizzazioni](#) nella Guida per l'utente di AWS IAM Identity Center.
- **Autorizzazioni utente IAM temporanee:** un utente IAM o un ruolo può assumere un ruolo IAM per ottenere temporaneamente autorizzazioni diverse per un'attività specifica.
- **Accesso multi-account:** è possibile utilizzare un ruolo IAM per permettere a un utente (un principale affidabile) con un account diverso di accedere alle risorse nell'account. I ruoli sono lo strumento principale per concedere l'accesso multi-account. Tuttavia, per alcuni dei Servizi AWS, è possibile collegare una policy direttamente a una risorsa (anziché utilizzare un ruolo come proxy). Per

informazioni sulle differenze tra ruoli e policy basate su risorse per l'accesso multi-account, consulta [Differenza tra i ruoli IAM e le policy basate su risorse](#) nella Guida per l'utente di IAM.

- **Accesso multi-servizio:** alcuni Servizi AWS utilizzano funzionalità in altri Servizi AWS. Ad esempio, quando effettui una chiamata in un servizio, è comune che tale servizio esegua applicazioni in Amazon EC2 o archivi oggetti in Amazon S3. Un servizio può eseguire questa operazione utilizzando le autorizzazioni dell'entità chiamante, utilizzando un ruolo di servizio o utilizzando un ruolo collegato al servizio.
- **Inoltro delle sessioni di accesso (FAS):** quando si utilizza un utente o un ruolo IAM per eseguire operazioni in AWS, tale utente o ruolo viene considerato un principale. Quando si utilizzano alcuni servizi, è possibile eseguire un'operazione che attiva un'altra azione in un servizio diverso. FAS utilizza le autorizzazioni del principale che effettua la chiamata a un Servizio AWS, combinate con il Servizio AWS richiedente, per effettuare richieste a servizi a valle. Le richieste FAS vengono effettuate solo quando un servizio riceve una richiesta che necessita di interazioni con altri Servizi AWS o risorse per essere portata a termine. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le operazioni. Per i dettagli delle policy relative alle richieste FAS, consulta la pagina [Forward access sessions](#).
- **Ruolo di servizio:** un ruolo di servizio è un [ruolo IAM](#) assunto da un servizio per eseguire operazioni per conto dell'utente. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta la sezione [Creazione di un ruolo per delegare le autorizzazioni a un Servizio AWS](#) nella Guida per l'utente di IAM.
- **Ruolo collegato al servizio:** un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un Servizio AWS. Il servizio può assumere il ruolo per eseguire un'azione per tuo conto. I ruoli collegati ai servizi sono visualizzati nell'account Account AWS e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati ai servizi, ma non modificarle.
- **Applicazioni in esecuzione su Amazon EC2:** è possibile utilizzare un ruolo IAM per gestire credenziali temporanee per le applicazioni in esecuzione su un'istanza EC2 che eseguono richieste di AWS CLI o dell'API AWS. Ciò è preferibile all'archiviazione delle chiavi di accesso nell'istanza EC2. Per assegnare un ruolo AWS a un'istanza EC2, affinché sia disponibile per tutte le relative applicazioni, puoi creare un profilo dell'istanza collegato all'istanza. Un profilo dell'istanza contiene il ruolo e consente ai programmi in esecuzione sull'istanza EC2 di ottenere le credenziali temporanee. Per ulteriori informazioni, consulta [Utilizzo di un ruolo IAM per concedere autorizzazioni ad applicazioni in esecuzione su istanze di Amazon EC2](#) nella Guida per l'utente di IAM.

Per informazioni sull'utilizzo dei ruoli IAM, consulta [Quando creare un ruolo IAM \(invece di un utente\)](#) nella Guida per l'utente di IAM.

Gestione dell'accesso con policy

Per controllare l'accesso a AWS è possibile creare policy e collegarle a identità o risorse AWS. Una policy è un oggetto in AWS che, quando associato a un'identità o a una risorsa, ne definisce le autorizzazioni. AWS valuta queste policy quando un principale IAM (utente, utente root o sessione ruolo) effettua una richiesta. Le autorizzazioni nelle policy determinano l'approvazione o il rifiuto della richiesta. La maggior parte delle policy viene archiviata in AWS sotto forma di documenti JSON. Per ulteriori informazioni sulla struttura e sui contenuti dei documenti delle policy JSON, consulta [Panoramica delle policy JSON](#) nella Guida per l'utente di IAM.

Gli amministratori possono utilizzare le policy AWSJSON per specificare l'accesso ai diversi elementi. In altre parole, quale principale può eseguire azioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Per concedere agli utenti l'autorizzazione a eseguire azioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM. Successivamente l'amministratore può aggiungere le policy IAM ai ruoli e gli utenti possono assumere i ruoli.

Le policy IAM definiscono le autorizzazioni relative a un'operazione, a prescindere dal metodo utilizzato per eseguirla. Ad esempio, supponiamo di disporre di una policy che consente l'azione `iam:GetRole`. Un utente con tale policy può ottenere informazioni sul ruolo dalla AWS Management Console, la AWS CLI o l'API AWS.

Policy basate su identità

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruolo IAM). Tali policy definiscono le azioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta [Creazione di policy IAM](#) nella Guida per l'utente di IAM.

Le policy basate su identità possono essere ulteriormente classificate come policy inline o policy gestite. Le policy inline sono incorporate direttamente in un singolo utente, gruppo o ruolo. Le policy gestite sono policy autonome che possono essere collegate a più utenti, gruppi e ruoli in Account AWS. Le policy gestite includono le policy gestite da AWS e le policy gestite dal cliente. Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta [Scelta fra policy gestite e policy inline](#) nella Guida per l'utente di IAM.

Policy basate su risorse

Le policy basate su risorse sono documenti di policy JSON che è possibile allegare a una risorsa. Gli esempi più comuni di policy basate su risorse sono le policy di attendibilità dei ruoli IAM e le policy dei bucket Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarle per controllare l'accesso a una risorsa specifica. Quando è allegata a una risorsa, una policy definisce le azioni che un principale può eseguire su tale risorsa e a quali condizioni. È necessario [specificare un principale](#) in una policy basata sulle risorse. I principali possono includere account, utenti, ruoli, utenti federati o Servizi AWS.

Le policy basate sulle risorse sono policy inline che si trovano in tale servizio. Non è possibile utilizzare le policy gestite da AWS provenienti da IAM in una policy basata su risorse.

Liste di controllo degli accessi (ACL)

Le liste di controllo degli accessi (ACL) controllano quali principali (membri, utenti o ruoli dell'account) hanno le autorizzazioni per accedere a una risorsa. Le ACL sono simili alle policy basate sulle risorse, sebbene non utilizzino il formato del documento di policy JSON.

Amazon S3, AWS WAF e Amazon VPC sono esempi di servizi che supportano le ACL. Per maggiori informazioni sulle ACL, consulta [Panoramica delle liste di controllo degli accessi \(ACL\)](#) nella Guida per gli sviluppatori di Amazon Simple Storage Service.

Altri tipi di policy

AWS supporta altri tipi di policy meno comuni. Questi tipi di policy possono impostare il numero massimo di autorizzazioni concesse dai tipi di policy più comuni.

- **Limiti delle autorizzazioni:** un limite delle autorizzazioni è una funzione avanzata nella quale si imposta il numero massimo di autorizzazioni che una policy basata su identità può concedere a un'entità IAM (utente o ruolo IAM). È possibile impostare un limite delle autorizzazioni per un'entità. Le autorizzazioni risultanti sono l'intersezione delle policy basate su identità dell'entità e i relativi limiti delle autorizzazioni. Le policy basate su risorse che specificano l'utente o il ruolo nel campo `Principal` sono condizionate dal limite delle autorizzazioni. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni sui limiti delle autorizzazioni, consulta [Limiti delle autorizzazioni per le entità IAM](#) nella Guida per l'utente di IAM.
- **Policy di controllo dei servizi (SCP):** le SCP sono policy JSON che specificano il numero massimo di autorizzazioni per un'organizzazione o unità organizzativa (OU) in AWS Organizations. AWS Organizations è un servizio per il raggruppamento e la gestione centralizzata degli Account

AWSmultipli di proprietà dell'azienda. Se abiliti tutte le funzionalità in un'organizzazione, puoi applicare le policy di controllo dei servizi (SCP) a uno o tutti i tuoi account. La SCP limita le autorizzazioni per le entità negli account membri, compreso ogni Utente root dell'account AWS. Per ulteriori informazioni su organizzazioni e policy SCP, consulta la pagina sulle [Policy di controllo dei servizi](#) nella Guida per l'utente di AWS Organizations.

- **Policy di sessione:** le policy di sessione sono policy avanzate che vengono trasmesse come parametro quando si crea in modo programmatico una sessione temporanea per un ruolo o un utente federato. Le autorizzazioni della sessione risultante sono l'intersezione delle policy basate su identità del ruolo o dell'utente e le policy di sessione. Le autorizzazioni possono anche provenire da una policy basata su risorse. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni, consulta [Policy di sessione](#) nella Guida per l'utente di IAM.

Più tipi di policy

Quando più tipi di policy si applicano a una richiesta, le autorizzazioni risultanti sono più complicate da comprendere. Per informazioni su come AWS determina se consentire una richiesta quando sono coinvolti più tipi di policy, consulta [Logica di valutazione delle policy](#) nella Guida per l'utente di IAM.

Funzionamento dei Servizi AWS con IAM

Per ottenere un quadro generale del funzionamento dei Servizi AWS con la maggior parte delle funzionalità di IAM, consulta la pagina [AWS services that work with IAM](#) della Guida per l'utente IAM.

Per ulteriori informazioni su come utilizzare uno specifico Servizio AWS con IAM, consulta la sezione sulla sicurezza della guida per l'utente del servizio pertinente.

Risoluzione dei problemi di identità e accesso in AWS

Utilizza le informazioni seguenti per diagnosticare e risolvere i problemi comuni che possono verificarsi durante l'utilizzo di AWS e di IAM.

Argomenti

- [Non sono autorizzato a eseguire un'operazione in AWS](#)
- [Non sono autorizzato a eseguire iam:PassRole](#)
- [Voglio consentire alle persone esterne al mio account Account AWSdi accedere alle mie risorse AWS](#)

Non sono autorizzato a eseguire un'operazione in AWS

Se ricevi un errore che indica che non sei autorizzato a eseguire un'operazione, le tue policy devono essere aggiornate per poter eseguire l'operazione.

L'errore di esempio seguente si verifica quando l'utente IAM `mateojackson` prova a utilizzare la console per visualizzare i dettagli relativi a una risorsa `my-example-widget` fittizia ma non dispone di autorizzazioni `awes:GetWidget` fittizie.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:  
awes:GetWidget on resource: my-example-widget
```

In questo caso, la policy per l'utente `mateojackson` deve essere aggiornata per consentire l'accesso alla risorsa `my-example-widget` utilizzando l'azione `awes:GetWidget`.

Per ulteriore assistenza con l'accesso, contatta l'amministratore AWS. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Non sono autorizzato a eseguire iam:PassRole

Se ricevi un errore che indica che non sei autorizzato a eseguire l'operazione `iam:PassRole`, le tue policy devono essere aggiornate per poter passare un ruolo a AWS.

Alcuni Servizi AWS consentono di passare un ruolo esistente a tale servizio, invece di creare un nuovo ruolo di servizio o un ruolo collegato ai servizi. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per trasmettere il ruolo al servizio.

L'errore di esempio seguente si verifica quando un utente IAM denominato `marymajor` cerca di utilizzare la console per eseguire un'operazione in AWS. Tuttavia, l'operazione richiede che il servizio disponga delle autorizzazioni concesse da un ruolo di servizio. Mary non dispone delle autorizzazioni per passare il ruolo al servizio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:  
iam:PassRole
```

In questo caso, le policy di Mary devono essere aggiornate per poter eseguire l'operazione `iam:PassRole`.

Per ulteriore assistenza con l'accesso, contatta l'amministratore AWS. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Voglio consentire alle persone esterne al mio account Account AWSdi accedere alle mie risorse AWS

È possibile creare un ruolo con il quale utenti in altri account o persone esterne all'organizzazione possono accedere alle tue risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per servizi che supportano policy basate su risorse o liste di controllo accessi (ACL), utilizza tali policy per concedere alle persone l'accesso alle tue risorse.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- Per capire se AWS supporta queste funzionalità, consulta [Funzionamento dei Servizi AWS con IAM](#).
- Per informazioni su come garantire l'accesso alle risorse negli Account AWSche possiedi, consulta [Fornire l'accesso a un utente IAM in un altro Account AWS in tuo possesso](#) nella Guida per l'utente IAM.
- Per informazioni su come fornire l'accesso alle risorse ad Account AWS di terze parti, consulta [Fornire l'accesso agli Account AWS di proprietà di terze parti](#) nella Guida per l'utente IAM.
- Per informazioni su come fornire l'accesso tramite la federazione delle identità, consulta [Fornire l'accesso a utenti autenticati esternamente \(Federazione delle identità\)](#) nella Guida per l'utente di IAM.
- Per informazioni sulle differenze tra l'utilizzo di ruoli e policy basate su risorse per l'accesso multi-account, consultare [Differenza tra i ruoli IAM e le policy basate su risorse](#) nella Guida per l'utente di IAM.

Convalida della conformità per questo prodotto o servizio AWS

Per sapere se il Servizio AWSè coperto da programmi di conformità specifici, consulta i [Servizi AWScoperti dal programma di conformità](#) e scegli il programma di conformità desiderato. Per informazioni generali, consulta [Programmi per la conformità di AWS](#).

È possibile scaricare i report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta [Download di report in AWS Artifact](#).

La responsabilità di conformità durante l'utilizzo dei Servizi AWSè determinata dalla riservatezza dei dati, dagli obiettivi di conformità dell'azienda e dalle normative vigenti. Per semplificare il rispetto della conformità, AWSmette a disposizione le seguenti risorse:

- [Guide Quick Start per la sicurezza e conformità](#): queste guide all'implementazione illustrano considerazioni relative all'architettura e forniscono la procedura per l'implementazione di ambienti di base su AWS incentrati sulla sicurezza e sulla conformità.
- [Architetture per la sicurezza e la conformità HIPAA su Amazon Web Services](#): questo whitepaper descrive come le aziende possono utilizzare AWS per creare applicazioni conformi alla normativa HIPAA.

Note

Non tutti i Servizi AWS sono conformi ai requisiti HIPAA. Per ulteriori informazioni, consulta la sezione [Riferimenti sui servizi conformi ai requisiti HIPAA](#).

- [Risorse per la conformità AWS](#): una raccolta di cartelle di lavoro e guide suddivise per settore e area geografica.
- [AWS Guide alla conformità dei clienti](#): comprendi il modello di responsabilità condivisa attraverso la lente della conformità. Le guide riassumono le migliori pratiche per la protezione Servizi AWS e mappano le linee guida per i controlli di sicurezza su più framework (tra cui il National Institute of Standards and Technology (NIST), il Payment Card Industry Security Standards Council (PCI) e l'International Organization for Standardization (ISO)).
- [Valutazione delle risorse con le regole](#) nella Guida per gli sviluppatori di AWS Config: il servizio AWS Config valuta il livello di conformità delle configurazioni delle risorse con pratiche interne, linee guida e regolamenti.
- [AWS Security Hub](#): questo Servizio AWS fornisce una visione completa dello stato di sicurezza all'interno di AWS. La Centrale di sicurezza utilizza i controlli di sicurezza per valutare le risorse AWS e verificare la conformità agli standard e alle best practice del settore della sicurezza. Per un elenco dei servizi e dei controlli supportati, consulta la pagina [Documentazione di riferimento sui controlli della Centrale di sicurezza](#).
- [AWS Audit Manager](#): questo Servizio AWS aiuta a verificare continuamente l'utilizzo di AWS per semplificare la gestione dei rischi e della conformità alle normative e agli standard di settore.

Questo prodotto o servizio AWS segue il [modello di responsabilità condivisa](#) attraverso gli specifici servizi Amazon Web Services (AWS) supportati. Per informazioni sulla sicurezza del servizio AWS, consulta la [pagina relativa alla documentazione della sicurezza dei servizi AWS](#) e i [Servizi AWS coperti dal programma di compliance che rientrano nell'ambito delle attività di conformità AWS in base al programma di compliance](#).

Applicazione di una versione minima di TLS in Strumenti per PowerShell

Per aumentare la sicurezza durante la comunicazione con i servizi AWS, devi configurare Strumenti per PowerShell in modo che venga utilizzata la versione di TLS appropriata. Per informazioni su come eseguire questa operazione, consulta [Enforcing a minimum TLS version](#) (Applicazione di una versione minima di TLS) nella [Guida per gli sviluppatori di AWS SDK for .NET](#).

Considerazioni aggiuntive sulla sicurezza per gli strumenti per PowerShell

Questo argomento contiene considerazioni sulla sicurezza oltre agli argomenti sulla sicurezza trattati nelle sezioni precedenti.

Registrazione di informazioni sensibili

Alcune operazioni di questo strumento potrebbero restituire informazioni che potrebbero essere considerate sensibili, incluse le informazioni provenienti da variabili di ambiente. L'esposizione di queste informazioni potrebbe rappresentare un rischio per la sicurezza in determinati scenari; ad esempio, le informazioni potrebbero essere incluse nei registri di integrazione e distribuzione continua (CI/CD). È quindi importante verificare quando includere tale output nei log e sopprimerlo quando non è necessario. Per ulteriori informazioni sulla protezione dei dati sensibili, vedere [Protezione dei dati in questo prodotto o servizio AWS](#)

Prendi in considerazione le seguenti best practice:

- Non utilizzate variabili di ambiente per memorizzare valori sensibili per le vostre risorse serverless. Fate invece in modo che il codice serverless recuperi a livello di codice il segreto da un archivio di segreti (ad esempio,). AWS Secrets Manager
- Esamina il contenuto dei log di compilazione per assicurarti che non contengano informazioni riservate. Prendi in considerazione approcci come il piping verso `/dev/null` o l'acquisizione dell'output come `bash` o variabile per sopprimere gli output dei comandi. PowerShell
- Prendi in considerazione l'accesso ai tuoi log e definisci l'accesso in modo appropriato per il tuo caso d'uso.

Cmdlet reference for the Tools for PowerShell

Tools for PowerShell fornisce cmdlet che puoi utilizzare per accedere ai servizi AWS. Per vedere quali cmdlet sono disponibili, consulta [AWS Tools for PowerShell Cmdlet Reference](#).

Cronologia dei documenti

Questo argomento descrive notevoli modifiche alla documentazione per gli AWS Tools for PowerShell.

Abbiamo inoltre aggiornato la documentazione periodicamente in risposta ai feedback dei clienti. Per inviare feedback su un argomento, utilizza i pulsanti di feedback accanto alla domanda "Questa pagina ti è stata utile?" nella parte inferiore di ogni pagina.

[Per ulteriori informazioni sulle modifiche e gli aggiornamenti di AWS Tools for PowerShell, consulta le note di rilascio.](#)

Modifica	Descrizione	Data
Codici di esempio	È incluso un capitolo con esempi di cmdlet.	17 aprile 2024
Considerazioni aggiuntive sulla sicurezza	Informazioni incluse sulla potenziale registrazione di dati sensibili.	16 aprile 2024
Configura l'autenticazione dello strumento con AWS	Sono state aggiunte informazioni sul supporto per SSO in AWS Tools for PowerShell	15 marzo 2024
Riferimento al cmdlet per il programma Tools for PowerShell	È stata aggiunta una sezione con un collegamento al riferimento Tools for PowerShell cmdlet.	17 novembre 2023
Sono stati inclusi ulteriori aggiornamenti delle best practice IAM	Guida aggiornata per l'allineamento alle best practice IAM. Per ulteriori informazioni, consulta Best practice per la sicurezza in IAM .	12 ottobre 2023
Installazione su Windows	Sono state rimosse le informazioni sull'installazione	25 settembre 2023

	degli strumenti per Windows PowerShell utilizzando l'MSI, che è diventato obsoleto.	
Aggiornamenti delle best practice di IAM	Guida aggiornata per l'allineamento alle best practice IAM. Per ulteriori informazioni, consulta Best practice per la sicurezza in IAM .	8 settembre 2023
Pipelining e \$ AWSHistory	È stato aggiunto il parametro IncludeSensitiveData al cmdlet Set-AWSHistoryConfiguration .	9 marzo 2023
Utilizzo del ClientConfig parametro nei cmdlet	Sono state aggiunte informazioni sul supporto per il ClientConfig parametro.	28 ottobre 2022
Avvio di un'istanza Amazon EC2 tramite Windows PowerShell	Aggiunte note relative al ritiro di EC2-Classic.	26 luglio 2022
AWS Tools for PowerShell Versione 4	Sono state aggiunte informazioni sulla versione 4, incluse le istruzioni di installazione per Windows e Linux/macOS , e un argomento di migrazione che descrive le differenze rispetto alla versione 3 e introduce nuove funzionalità.	21 novembre 2019

[AWS Tools for PowerShell](#)
[3.3.563](#)

Sono state aggiunte informazioni su come installare e utilizzare la versione di anteprima del modulo `AWS.Tools.Common`. Questo nuovo modulo suddivide il vecchio pacchetto monolitico in un modulo condiviso e un modulo per servizio. AWS

18 ottobre 2019

[AWS Tools for PowerShell](#)
[3.3.343.0](#)

Sono state aggiunte informazioni alla sezione [Utilizzo della AWS Tools for PowerShell](#) [I sezione che](#) introduce gli AWS Lambda strumenti per gli sviluppatori PowerShell Core PowerShell per creare funzioni. AWS Lambda

11 settembre 2018

[AWS Tools for Windows PowerShell 3.1.31.0](#)

Sono state aggiunte informazioni alla sezione [Nozioni di base](#) riguardo nuovi cmdlet che utilizzano Security Assertion Markup Language (SAML) per supportare la configurazione di identità federate per gli utenti.

1 dicembre 2015

[AWS Tools for Windows PowerShell 2,3,19](#)

Sono state aggiunte informazioni alla sezione [Cmdlets Discovery and Aliases](#) sul nuovo `Get-AWSCmdletName` cmdlet che possono aiutare gli utenti a trovare più facilmente i cmdlet desiderati. AWS

5 febbraio 2015

[AWS Tools for Windows PowerShell 1.1.1.0](#)

15 maggio 2013

L'output della raccolta dai cmdlet viene sempre enumerato nella pipeline. PowerShell È disponibile il supporto automatico per le chiamate di servizio impaginabili. La nuova variabile `$ AWSHistory` raccoglie le risposte di servizio e, facoltativamente, le richieste di assistenza. `AWSRegion` le istanze utilizzano il campo `Region` invece di `SystemName` per facilitare la pipelining. `Remove-S3 Buckets` supporta un'opzione `Delete Objects` - switch. Risolto il problema di usabilità con `Set-AWSCredentials`. Inizializza: `AWSDefaults` riporta da dove ha ottenuto le credenziali e i dati della regione. `Stop-EC2Instances` accetta istanze Amazon. `ec2.model.Reservation` come input. I tipi di parametro `Generic List<T>` sono sostituiti con i tipi di array (`T[]`). I cmdlet che eliminano o terminano le risorse, richiedono conferma prima dell'eliminazione. `Write-S3Objects` supporta contenuti di testo in linea da caricare su Amazon S3.

[AWS Tools for Windows PowerShell 1.0.1.0](#)

21 dicembre 2012

La posizione di installazione del PowerShell modulo Tools for Windows è stata modificata in modo che gli ambienti che utilizzano Windows PowerShell versione 3 possano sfruttare il caricamento automatico. Il modulo e i file di supporto sono ora installati in una sottocartella `AWSPowerShell` in `AWS ToolsPowerShell`. I file delle versioni precedenti esistenti nella cartella `AWS ToolsPowerShell` vengono rimossi automaticamente dal programma di installazione. Il `PSModulePath` per Windows PowerShell (tutte le versioni) viene aggiornato in questa versione per contenere la cartella principale del modulo (`AWS ToolsPowerShell`). Per i sistemi con Windows PowerShell versione 2, la scorciatoia del menu Start viene aggiornata per importare il modulo dalla nuova posizione e quindi eseguirlo `Initialize-AWSDefaults`. Per i sistemi con Windows PowerShell versione 3, la scorciatoia del menu Start viene aggiornata per rimuovere il `Import-Module` comando,

lasciando solo. `Initialize-AWSDefaults`. Se hai modificato il PowerShell I profilo per eseguire uno `Import-Module` dei `AWSPowerShell.psd1` file, dovrai aggiornarlo in modo che punti alla nuova posizione del file (oppure, se usi la PowerShell versione 3, rimuovere l'`Import-Module` istruzione poiché non è più necessaria). Come risultato di queste modifiche, il PowerShell modulo Tools for Windows è ora elencato come modulo disponibile durante l'esecuzione `Get-Module -ListAvailable`. Inoltre, per gli utenti di Windows PowerShell versione 3, l'esecuzione di qualsiasi cmdlet esportato dal modulo caricherà automaticamente il modulo nella PowerShell shell corrente senza doverlo prima utilizzare. `Import-Module` Questo abilita l'utilizzo interattivo dei cmdlet su un sistema con una policy di esecuzione che non consente l'esecuzione di script.

[AWS Tools for Windows PowerShell 1.0.0.0](#)

Rilascio iniziale

6 dicembre 2012

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.