

使用者指南

AWS Tools for PowerShell (第 4 版)



AWS Tools for PowerShell (第 4 版) : 使用者指南

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商標和商業外觀不得用於任何非 Amazon 的產品或服務，也不能以任何可能造成客戶混淆、任從何貶低或使 Amazon 名譽受損的方式使用 Amazon 的商標和商業外觀。所有其他非 Amazon 擁有的商標均為其各自擁有者的財產，這些擁有者可能附屬於 Amazon，或與 Amazon 有合作關係，亦或受到 Amazon 贊助。

Table of Contents

.....	x
AWS Tools for PowerShell是什麼？	1
開發套件主要版本的維護與支援	1
AWS.Tools	2
AWSPowerShell.NetCore	2
AWSPowerShell	3
本指南的使用方式	3
本區段的其他主題	4
修訂歷史記錄	4
新功能	4
安裝	6
在 Windows 上安裝	6
先決條件	7
安裝 AWS.Tools	7
安裝 AWSPowerShell.NetCore	9
安裝 AWSPowerShell	11
啟用指令碼執行	11
版本控制	13
更新 AWS Tools for PowerShell	15
在 Linux 或 macOS 上安裝	16
設定概觀	16
先決條件	7
安裝 AWS.Tools	17
安裝 AWSPowerShell.NetCore	19
指令碼執行	11
設定 PowerShell 主控台	21
初始化您的 PowerShell 工作階段	22
版本控制	13
在 Linux 或 macOS AWS Tools for PowerShell 上更新	23
相關資訊	24
從 3.3 AWS Tools for PowerShell 版遷移至 第 4 版	24
新全面模組化 AWS.Tools 版本	24
全新 Get-AWSService Cmdlet	25
控制 Cmdlet 傳回之物件的全新 -Select 參數	25

以更一致的方式限制輸出的項目數	27
更易於使用的串流參數	27
依屬性名稱擴充管道	28
靜態常見參數	28
AWS.Tools 會宣告並強制執行必要參數	29
所有參數皆可為 Null	29
移除先前取代的功能	29
開始使用	30
設定工具身分驗證	30
啟用和設定 IAM Identity Center	31
設定 Tools for PowerShell 以使用 IAM Identity Center。	31
啟動 AWS 存取入口網站工作階段	33
範例	33
其他資訊	34
使用 AWS CLI	34
指定 AWS 區域	37
指定自訂或非標準端點	39
其他資訊	39
設定聯合身分	40
先決條件	40
聯合身分使用者如何取得 AWS 服務 APIs 聯合存取權	40
SAML 支援如何在 中運作 AWS Tools for PowerShell	41
如何使用 PowerShell SAML 組態 Cmdlet	43
其他閱讀資料	47
Cmdlet 探索和別名	47
Cmdlet 探索	47
Cmdlet 命名和別名	53
管道、輸出和反覆運算	57
管道	57
Cmdlet 輸出	58
透過分頁資料反覆運算	59
憑證和設定檔解析	61
憑證搜尋順序	61
使用者和角色	62
使用者和許可集合	62
服務角色	62

使用舊版憑證	63
重要警告和指引	63
AWS 登入資料	64
共用憑證	72
功能	77
可觀測性	77
使用 AWS 服務	81
PowerShell 檔案串連編碼	81
Powershell 工具傳回的物件	81
Amazon EC2	82
Amazon Simple Storage Service (Amazon S3)	82
AWS Lambda 而且 AWS Tools for PowerShell	82
Amazon SNS 和 Amazon SQS	82
CloudWatch	83
另請參閱	83
主題	83
Amazon S3 和 Tools for Windows PowerShell	83
建立 Amazon S3 儲存貯體、驗證其區域，或者加以移除	84
設定 Amazon S3 儲存貯體做為網站並啟用記錄	85
將物件上傳至 Amazon S3 儲存貯體	86
刪除 Amazon S3 物件與儲存貯體	88
將內嵌文字內容上傳到 Amazon S3	89
Amazon EC2 與 Tools for Windows PowerShell	90
建立金鑰對	90
建立安全群組	92
尋找 AMI	95
啟動 執行個體	98
AWS Lambda 而且 AWS Tools for PowerShell	101
先決條件	7
安裝 AWSLambdaPSCore 模組	102
另請參閱	83
Amazon SQS、Amazon SNS 和 Tools for Windows PowerShell	102
建立 Amazon SQS 佇列和取得佇列 ARN	102
建立 Amazon SNS 主題	103
提供許可給 SNS 主題	103
訂閱佇列至 SNS 主題	104

提供許可	104
驗證結果	104
來自的 CloudWatch AWS Tools for Windows PowerShell	106
發布自訂指標至 CloudWatch 儀表板	106
另請參閱	83
使用 ClientConfig	106
使用 ClientConfig 參數。	107
使用未定義的屬性	107
指定 AWS 區域	108
程式碼範例	109
ACM	111
動作	111
Application Auto Scaling	115
動作	111
AppStream 2.0	122
動作	111
Aurora	149
動作	111
Auto Scaling	150
動作	111
AWS Budgets	185
動作	111
AWS Cloud9	186
動作	111
AWS CloudFormation	192
動作	111
CloudFront	204
動作	111
CloudTrail	211
動作	111
CloudWatch	216
動作	111
CodeCommit :	220
動作	111
CodeDeploy	226
動作	111

CodePipeline	244
動作	111
Amazon Cognito 身分	262
動作	111
AWS Config	266
動作	111
Device Farm	285
動作	111
AWS Directory Service	286
動作	111
AWS DMS	311
動作	111
DynamoDB	312
動作	111
Amazon EC2	326
動作	111
Amazon ECR	455
動作	111
Amazon ECS	456
動作	111
Amazon EFS	462
動作	111
Amazon EKS	469
動作	111
Elastic Load Balancing - 第 1 版	481
動作	111
Elastic Load Balancing - 第 2 版	501
動作	111
Amazon FSx	524
動作	111
AWS Glue	532
動作	111
AWS Health	534
動作	111
IAM	535
動作	111

Kinesis	606
動作	111
Lambda	609
動作	111
Amazon ML	622
動作	111
Macie	627
動作	111
AWS OpsWorks	628
動作	111
AWS 價格表	630
動作	111
資源群組	632
動作	111
Resource Groups 標記 API	640
動作	111
Route 53	645
動作	111
Amazon S3	659
動作	111
S3 Glacier	693
動作	111
安全中樞	697
動作	111
Amazon SES	698
動作	111
Amazon SES API v2	700
動作	111
Amazon SNS	700
動作	111
Amazon SQS	702
動作	111
AWS STS	714
動作	111
支援	718
動作	111

Systems Manager	723
動作	111
Amazon Translate	795
動作	111
AWS WAFV2	795
動作	111
WorkSpaces	796
動作	111
安全	812
資料保護	812
資料加密	813
身分和存取權管理	813
目標對象	814
使用身分驗證	814
使用政策管理存取權	817
AWS 服務 如何使用 IAM	819
對 AWS 身分和存取進行故障診斷	819
合規驗證	821
強制執行最低 TLS 版本	821
其他安全考慮事項	822
記錄敏感資訊	822
Cmdlet 參考	823
文件歷史紀錄	824
.....	dccccxxx

第 5 版 (V5) AWS Tools for PowerShell 已發行！

若要開始使用新版本的工具，請參閱 [AWS Tools for PowerShell 使用者指南 \(V5\)](#)，特別是[遷移至 V5](#) 的主題。

AWS Tools for PowerShell是什麼？

AWS Tools for PowerShell 是一組 PowerShell 模組，以公開的功能為基礎適用於 .NET 的 AWS SDK。AWS Tools for PowerShell 可讓您從 PowerShell 命令列編寫 AWS 資源操作的指令碼。

cmdlet 提供慣用的 PowerShell 體驗，即使使用各種 AWS 服務 HTTP 查詢 APIs 實作參數和處理結果。例如，AWS Tools for PowerShell 支援 PowerShell 管道的 cmdlet，也就是說，您可以將 PowerShell 物件傳入和傳出 cmdlet。

可讓您處理登入資料的方式 AWS Tools for PowerShell 相當靈活，包括對 AWS Identity and Access Management (IAM) 基礎設施的支援。您可以使用這些工具，搭配 IAM 使用者憑證、暫時安全字符和 IAM 角色。

AWS Tools for PowerShell 支援 SDK 支援的相同服務和 AWS 區域集。您可以在執行 Windows、Linux 或 macOS 作業系統 AWS Tools for PowerShell 的電腦上安裝。

Note

AWS Tools for PowerShell 第 4 版 (V4) 是 3.3 AWS Tools for PowerShell 版的回溯相容更新。此版本新增了大幅改善的項目，同時保持現有的 Cmdlet 行為。您現有的指令碼應在升級至 V4 後繼續運作，但我們建議您在升級之前徹底測試它們。如需 V4 中變更的詳細資訊，請參閱 [從 3.3 AWS Tools for PowerShell 版遷移至 第 4 版](#)。

AWS Tools for PowerShell 提供以下三種不同的套件：

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

開發套件主要版本的維護與支援

如需開發套件主要版本及其基礎相依性之維護與支援的相關資訊，請參閱《[AWS 開發套件及工具參考指南](#)》中的以下內容：

- [AWS SDKs和工具維護政策](#)
- [AWS SDKs和工具版本支援矩陣](#)

AWS.Tools - 的模組化版本 AWS Tools for PowerShell

[PowerShell Gallery](#) [AWS.Tools.Installer](#)[PowerShell Gallery](#) [AWS.Tools.Common](#)[ZIP Archive](#) [AWS.Tools](#)

此版本的 AWS Tools for PowerShell 是在生產環境中執行 PowerShell 的任何電腦的建議版本。因為此版本已模組化，因此您必須僅下載並載入您要使用之服務的模組。這樣可以縮減下載時間、記憶體使用量，並能在大多數情況下啟用 AWS.Tools Cmdlet 的自動匯入功能，而不需先手動呼叫 Import-Module。

這是最新版本的 AWS Tools for PowerShell 和 在所有支援的作業系統上執行，包括 Windows、Linux 和 macOS。此套件為每個 AWS 服務提供一個安裝模組、AWS.Tools.Installer、一個通用模組 AWS.Tools.Common、和一個模組，例如 AWS.Tools.EC2、AWS.Tools.IdentityManagement、AWS.Tools.S3 等。

AWS.Tools.Installer 模組提供 cmdlet，可讓您安裝、更新和移除每個 AWS 服務的模組。此模組中的 Cmdlet 會自動確保您擁有支援您要使用之模組所需的所有相依模組。

AWS.Tools.Common 模組提供適用於非服務專用的組態和驗證 cmdlet。若要使用 AWS 服務的 cmdlet，只需執行 命令即可。PowerShell 會自動為您要執行其 cmdlet AWS 的服務匯入 AWS.Tools.Common 模組和 模組。如果您使用 AWS.Tools.Installer 模組來安裝服務模組，就會自動安裝此模組。

您可以在執行中的 AWS Tools for PowerShell 電腦上安裝此版本的：

- Windows、Linux 或 macOS 版的 PowerShell Core 6.0 或更新版本。
- Windows 版的 Windows PowerShell 5.1 或更新版本及 .NET Framework 4.7.2 或更新版本。

在本指南中，必須明確只指出此版本時，我們會以模組名稱指稱該版本：*AWS.Tools*。

AWSPowerShell.NetCore - 的單一模組版本 AWS Tools for PowerShell

[PowerShell Gallery](#) [AWSPowerShell.NetCore](#)[ZIP Archive](#) [AWSPowerShell.NetCore](#)

此版本包含單一大型模組，其中包含所有 AWS 服務的支援。您必須先手動匯入此模組，才能使用它。

您可以在執行中的 AWS Tools for PowerShell 電腦上安裝此版本的：

- Windows、Linux 或 macOS 版的 PowerShell Core 6.0 或更新版本。
- Windows 版的 Windows PowerShell 3.0 或更新版本及 .NET Framework 4.7.2 或更新版本。

在本指南中，必須明確只指出此版本時，我們會以模組名稱指稱該版本：AWSPowerShell.NetCore。

AWSPowerShell -適用於 Windows PowerShell 的單一模組版本

PowerShell Gallery **AWSPowerShell**

ZIP Archive **AWSPowerShell**

此版本的 AWS Tools for PowerShell 與執行 Windows PowerShell 2.0 版到 5.1 版的 Windows 電腦相容且可安裝。它與 PowerShell Core 6.0 或更新版本，或任何其他作業系統 (Linux 或 macOS) 都不相容。此版本包含單一大型模組，其中包含所有 AWS 服務的支援。

在本指南中，必須明確只指出此版本時，我們會以模組名稱指稱該版本：AWSPowerShell。

本指南的使用方式

本指南分為以下幾個主要章節：

[安裝 AWS Tools for PowerShell](#)

本節說明如何安裝 AWS Tools for PowerShell。其中包括如何在您還沒有帳戶 AWS 時註冊，以及如何建立可用來執行 cmdlet 的 IAM 使用者。

[開始使用 AWS Tools for Windows PowerShell](#)

本節說明使用的基本概念 AWS Tools for PowerShell，例如指定登入資料和 AWS 區域、尋找特定服務的 cmdlet，以及使用 cmdlet 的別名。

[在中使用 AWS 服務 AWS Tools for PowerShell](#)

本節包含使用 AWS Tools for PowerShell 執行一些最常見 AWS 任務的相關資訊。

本區段的其他主題

- [修訂歷史記錄](#)
- [中的新功能 AWS Tools for PowerShell](#)

修訂歷史記錄

若要了解各種版本中的變更，請參閱以下內容：

- [變更日誌](#)
- [中的新功能 AWS Tools for PowerShell](#)
- [文件歷史紀錄](#)

中的新功能 AWS Tools for PowerShell

如需 相關新開發的高階資訊 AWS Tools for PowerShell，請參閱位於 <https://aws.amazon.com/powershell/> 的產品頁面和 [變更日誌](#)。

以下是 Tools for PowerShell 中的新功能。

2025 年 6 月 23 日：第 5 版 AWS Tools for PowerShell

第 5 版 (V5) AWS Tools for PowerShell 已全面推出！如需詳細資訊，請參閱 [AWS Tools for PowerShell 使用者指南 \(V5\)](#)，特別是[遷移至 V5](#) 的主題。

2025 年 2 月 10 日：GA 發行以獲得可觀測性

可觀測性是從系統發出的資料推斷系統目前狀態的程度。可觀測性已新增至 Tools for PowerShell，包括遙測供應商的實作。如需詳細資訊，請參閱本指南[可觀測性](#)中的 和部落格文章[宣布 AWS .NET OpenTelemetry 程式庫的一般可用性](#)。

2025 年 1 月 15 日：新的完整性保護預設行為

從 4.1.737 版開始 AWS Tools for PowerShell，這些工具會透過自動計算上傳的CRC32檢查總和來提供預設完整性保護。如需詳細資訊，請參閱 GitHub 上的公告，網址為 <https://github.com/aws/aws-tools-for-powershell/issues/370>。工具也提供資料完整性保護的全域設定，您可以在 [AWS SDKs 和工具參考指南](#)中的[資料完整性保護](#)中閱讀這些設定。

2024 年 11 月 18 日：第 5 版預覽版 1

第 5 AWS Tools for PowerShell 版的預覽版 1 已於 2024 年 11 月 18 日發行。如需此預覽的詳細資訊，請參閱部落格文章 [V5 的 AWS Tools for PowerShell 預覽 1](#)。

安裝 AWS Tools for PowerShell

若要成功安裝和使用 AWS Tools for PowerShell cmdlet，請參閱下列主題中的步驟。

主題

- [在 Windows AWS Tools for PowerShell 上安裝](#)
- [在 Linux 或 macOS AWS Tools for PowerShell 上安裝](#)
- [從 3.3 AWS Tools for PowerShell 版遷移至 第 4 版](#)

在 Windows AWS Tools for PowerShell 上安裝

Windows 型電腦可以執行任何 AWS Tools for PowerShell 套件選項：

- [AWS.Tools](#) - 的模組化版本 AWS Tools for PowerShell。每個 AWS 服務都由其個別的小型模組支援，具有共用支援模組 `AWS.Tools.Common` 和 `AWS.Tools.Installer`。
- [AWSPowerShell.NetCore](#) - 的單一大型模組版本 AWS Tools for PowerShell。此單一大型模組支援所有 AWS 服務。

Note

請注意，單一模組可能太大，無法與 [AWS Lambda](#) 功能一起使用。請改用上面的模組化版本。

- [AWSPowerShell](#) – AWS Tools for PowerShell 的舊式 Windows 專用單一大型模組版本。此單一大型模組支援所有 AWS 服務。

您選擇的套件取決於您正在執行的 Windows 版本。

Note

預設 AWS Tools for PowerShell 會在所有 Windows 型 Amazon Machine Image (AMIs) 上安裝。已安裝的選項取決於 AMI。許多 AMIs 都有 `AWSPowerShell` 模組，但有些模組可能有不同的選項。例如，適用於 Windows Server 2025 的 Amazon EC2 AMIs 使用模組化 `AWS.Tools` 選項。

設定 AWS Tools for PowerShell 涉及下列高階任務，在本主題中詳細說明。

1. 安裝適合您環境的 AWS Tools for PowerShell 套件選項。
2. 執行 Get-ExecutionPolicy Cmdlet 來驗證指令碼執行已啟用。
3. 將 AWS Tools for PowerShell 模組匯入 PowerShell 工作階段。

先決條件

若要從 Microsoft 取得更新版本的 PowerShell (包括 PowerShell Core)，請前往 Microsoft 網站的[安裝各種版本的 PowerShell](#) 進行下載。

在 Windows 上安裝 AWS.Tools

您可以在執行 Windows 搭配 Windows PowerShell 5 AWS Tools for PowerShell .1 或 PowerShell Core 6.0 或更新版本的電腦上安裝模組化版本的。如需如何安裝 PowerShell Core 的相關資訊，請參閱 Microsoft 網站上的[安裝各種版本的 PowerShell](#) 一文。

有三種方式可以安裝 AWS.Tools：

- 使用 AWS.Tools.Installer 模組中的 Cmdlet。此模組可簡化其他 AWS.Tools 模組的安裝和更新。AWS.Tools.Installer 需要 PowerShellGet，並會自動下載和安裝其更新版本。AWS.Tools.Installer 會自動讓您的模組版本保持同步。當您安裝或更新至某個模組的較新版本時，中的 cmdlet AWS.Tools.Installer 會自動將所有其他 AWS.Tools 模組更新為相同的版本。

此方法會在以下程序中說明。

- 從 [AWS.Tools.zip](#) 下載模組，然後解壓縮至其中一個模組資料夾。您可以顯示 PSModulePath 環境變數的值來探索自己的模組資料夾。

Warning

下載 ZIP 檔案之後，在擷取內容之前，您可能需要將其解鎖。這通常透過開啟 檔案的屬性、檢視一般索引標籤，以及在存在時選取解除封鎖核取方塊來完成。

如果需要解除封鎖 ZIP 檔案，但您未這麼做，您可能會收到類似下列的錯誤：「匯入模組：無法載入檔案或組件」。

- 使用 Install-Module cmdlet 從 PowerShell Gallery 安裝每個服務模組。

使用 **AWS.Tools.Installer** 模組在 Windows **AWS.Tools** 上安裝

1. 啟動 PowerShell 工作階段。

Note

除非手邊任務需要，否則建議「請勿」以提高權限的管理員身分執行 PowerShell。這是因為可能有潛在的安全風險，且不符合最低權限原則。

2. 若要安裝模組化 AWS.Tools 套件，請執行以下命令。

```
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
```

若您收到通知，顯示儲存庫「不受信任」，表示系統正在詢問您是否無論如何都要進行安裝。輸入 **y** 來允許 PowerShell 安裝模組。若要在不信任儲存庫的情況下避免出現提示及安裝模組，您可以執行命令搭配 **-Force** 參數。

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

3. 您現在可以使用 **Install-AWSToolsModule** cmdlet 為要使用的每個 AWS 服務安裝 模組。例如，以下命令會安裝 Amazon EC2 和 Amazon S3 模組。此命令也會安裝指定模組運作所需的任何相依模組。例如，當您安裝第一個 AWS.Tools 服務模組時，它也會安裝 **AWS.Tools.Common**。這是所有 AWS 服務模組所需的共用模組。它也會移除舊版模組，並將其他模組更新至相同的更新版本。

```
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

Install-AWSToolsModule Cmdlet 會從名為 PSGallery 的 PSRepository (<https://www.powershellgallery.com/>) 下載所有要求的模組，並視為是信任的來源。如需此 PSRepository 的詳細資訊，請使用 Get-PSRepository -Name PSGallery 命令。

根據預設，前一個命令會將模組安裝至 %USERPROFILE%\Documents\WindowsPowerShell\Modules 資料夾中。若要為某部電腦的所有使用者安裝 AWS Tools for PowerShell，您必須在您以管理員身分啟動的 PowerShell 工作階段中執行以下命令。例如，以下命令會將 IAM 模組安裝至所有使用者可存取的 %ProgramFiles%\WindowsPowerShell\Modules 資料夾。

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

若要安裝其他模組，請使用適當的模組名稱執行類似的命令，例如 [PowerShell Gallery](#) 中的命令。

在 Windows 上安裝 AWSPowerShell.NetCore

您可以在執行 Windows，且包含 PowerShell 第 3 版到 5.1 版，或是 PowerShell Core 6.0 或更新版本的電腦上安裝 AWSPowerShell.NetCore。如需如何安裝 PowerShell Core 的相關資訊，請參閱 Microsoft PowerShell 網站中的《[安裝各種版本的 PowerShell](#)》一文。

您可以透過下列兩種方法安裝 AWSPowerShell.NetCore

- 從 [AWSPowerShell.NetCore.zip](#) 下載模組，然後解壓縮至其中一個模組目錄。您可以顯示 PSModulePath 環境變數的值來探索自己的模組目錄。

Warning

下載 ZIP 檔案之後，在擷取內容之前，您可能需要將其解鎖。這通常是透過開啟 檔案的屬性、檢視一般索引標籤，以及在存在時選取解除封鎖核取方塊來完成。

如果需要解除封鎖 ZIP 檔案，但您未這麼做，您可能會收到類以下列的錯誤：「匯入模組：無法載入檔案或組件」。

- 使用 Install-Module Cmdlet 從 PowerShell Gallery 安裝，如下列程序中所描述。

使用 Install-Module cmdlet 從 PowerShell Gallery 安裝 AWSPowerShell.NetCore

若要從 PowerShell Gallery 安裝 AWSPowerShell.NetCore，您的電腦必須執行 PowerShell 5.0 或更新版本，或是在 PowerShell 3 或更新版本中執行 [PowerShellGet](#)。執行下列命令。

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

如果您以管理員身分執行 PowerShell，先前的命令會 AWS Tools for PowerShell 為電腦上的所有使用者安裝。如果您以沒有管理員許可的標準使用者身分執行 PowerShell，則只會 AWS Tools for PowerShell 為目前的使用者安裝相同的命令。

若在使用者具備管理員許可時，僅要為目前的使用者進行安裝，請搭配 -Scope CurrentUser 參數執行命令，如下。

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

雖然 PowerShell 3.0 和更新版本通常會在您第一次執行模組中的 Cmdlet 時，將模組載入您的 PowerShell 工作階段，由於 AWSPowerShell.NetCore 模組的大小太大，因此不支援此功能。您必須改為執行以下命令，明確將 AWSPowerShell.NetCore Core 模組載入您的 PowerShell 工作階段。

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

若要自動將 AWSPowerShell.NetCore 模組載入 PowerShell 工作階段，請將該命令新增到您的 PowerShell 描述檔。如需編輯您 PowerShell 描述檔的詳細資訊，請參閱 PowerShell 文件中的 [About Profiles](#)。

在 Windows PowerShell 上安裝 AWSPowerShell

您可以透過下列兩種方式 AWS Tools for Windows PowerShell 之一安裝：

- 從 [AWSPowerShell.zip](#) 下載模組，然後解壓縮至其中一個模組目錄。您可以顯示 PSModulePath 環境變數的值來探索自己的模組目錄。

Warning

下載 ZIP 檔案之後，在擷取內容之前，您可能需要將其解鎖。這通常是透過開啟檔案的屬性、檢視一般索引標籤，以及在存在時選取解除封鎖核取方塊來完成。

如果需要解除封鎖 ZIP 檔案，但您未這麼做，您可能會收到類以下列的錯誤：「匯入模組：無法載入檔案或組件」。

- 使用 Install-Module Cmdlet 從 PowerShell Gallery 安裝，如下列程序中所描述。

使用 Install-Module cmdlet 從 PowerShell Gallery 安裝 AWSPowerShell

如果您正在執行 PowerShell 5.0 或更新版本，或是已在 PowerShell 3 或更新版本上安裝 [PowerShellGet](#)，可以從 PowerShell Gallery 安裝 AWSPowerShell。您可以執行以下命令，從 Microsoft 的 [PowerShell Gallery](#) 安裝並更新 SWSPowerShell。

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

若要將 AWSPowerShell 模組自動載入 PowerShell 工作階段，請將先前的 import-module Cmdlet 新增到您的 PowerShell 描述檔。如需編輯您 PowerShell 描述檔的詳細資訊，請參閱 PowerShell 文件中的 [About Profiles](#)。

Note

Tools for Windows PowerShell 根據預設會安裝在所有 Windows 類型的 Amazon Machine Image (AMI) 上。

啟用指令碼執行

若要載入 AWS Tools for PowerShell 模組，您必須啟用 PowerShell 指令碼執行。欲啟用指令碼執行，請執行 Set-ExecutionPolicy cmdlet 來設定 RemoteSigned 政策。如需詳細資訊，請參閱 Microsoft Technet 網站上的 [About Execution Policies](#)。

Note

此需求僅適用於執行 Windows 的電腦。ExecutionPolicy 安全限制不會出現在其他作業系統上。

啟用指令碼執行

1. 需要管理員權限才能設定執行政策。若您並非以具備管理員權限的使用者登入，請以管理員身分開啟 PowerShell 工作階段。選擇 Start (開始)，然後選擇 All Programs (所有程式)。選擇 Accessories (附屬應用程式)，然後選擇 Windows PowerShell。以滑鼠右鍵按一下 Windows PowerShell，然後在內容選單上，選擇 Run as administrator (以系統管理員身分執行)。
2. 在命令提示中，輸入以下內容。

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

Note

在 64 位元系統上，您必須為 32 位元版本的 PowerShell (Windows PowerShell (x86)) 個別執行此作業。

若您沒有正確設定執行政策，PowerShell 會在您嘗試執行指令碼 (例如您的描述檔) 時顯示以下錯誤。

```
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
```

Tools for Windows PowerShell 安裝程式會自動更新 [PSModulePath](#)，納入內含 AWSPowerShell 模組的目錄位置。

由於 PSModulePath 包含 AWS 模組目錄的位置，因此 `Get-Module -ListAvailable` cmdlet 會顯示模組。

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

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

版本控制

AWS AWS Tools for PowerShell 會定期發行 的新版本，以支援新的 AWS 服務和功能。若要確認您已安裝的工具版本，請執行 [Get-AWSPowerShellVersion](#) Cmdlet。

例如：

```
PS > Get-AWSPowerShellVersion
```

```
AWS Tools for PowerShell  
Version 4.1.849  
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
Amazon Web Services SDK for .NET  
Core Runtime Version 3.7.402.75  
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
Release notes: https://github.com/aws/aws-tools-for-powershell/blob/v4.1/changelogs/  
CHANGELOG.ALL.md
```

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

您也可以將 `-ListServiceVersionInfo` 參數新增至 [Get-AWSPowerShellVersion](#) 命令，以查看目前版本工具中支援的 AWS 服務清單。若您使用模組化的 `AWS.Tools.*` 選項，則只會顯示您目前已匯入的模組。

例如：

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

Service                                Noun Prefix Module Name                                SDK
-----
Assembly
Version
-----
-----
AWS IAM Access Analyzer                IAMAA      AWS.Tools.AccessAnalyzer
3.7.400.33
AWS Account                           ACCT       AWS.Tools.Account
3.7.400.33
AWS Certificate Manager Private... PCA        AWS.Tools.ACMPCA
3.7.400.34
AWS Amplify                           AMP        AWS.Tools.Amplify
3.7.401.28
Amplify Backend                       AMPB       AWS.Tools.AmplifyBackend
3.7.400.33
...
```

若要確認您執行的 PowerShell 版本，請輸入 `$PSVersionTable` 來檢視 `$PSVersionTable` [自動變數](#)的內容。

例如：

```
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
```


在 Windows AWS Tools for PowerShell 上更新

定期 AWS Tools for PowerShell 發行更新版本的 時，您應該更新在本機執行的版本。

更新模組化 **AWS.Tools** 模組

若要將 AWS.Tools 模組更新至最新版本，請執行下列命令：

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

此命令會更新所有目前安裝的 AWS.Tools 模組，並且會在更新成功後移除其他安裝的版本。

Note

Update-AWSToolsModule Cmdlet 會從名為 PSGallery 的 PSRepository (<https://www.powershellgallery.com/>) 下載所有模組，並視為是信任的來源。如需此 PSRepository 的詳細資訊，請使用 Get-PSRepository -Name PSGallery 命令。

更新 Tools for PowerShell Core

執行 Get-AWSPowerShellVersion Cmdlet 來確認您正在執行的版本，並將此版本與 [PowerShell Gallery](https://www.powershellgallery.com/) 網站上可用的 Tools for Windows PowerShell 版本進行比較。我們建議您每兩到三週檢查一次。只有在您更新到具有該支援的版本之後，才能支援新的命令 AWS 和服務。

安裝 AWSPowerShell.NetCore 的較新版本前，請解除安裝現有模組。請在您解除安裝現有套件前關閉任何開啟的 PowerShell 工作階段。執行下列命令以解除安裝套件。

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

解除安裝套件後，請執行以下命令來安裝更新後的模組。

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

安裝完成後，請執行 Import-Module AWSPowerShell.NetCore 命令來將更新後的 Cmdlet 載入您的 PowerShell 工作階段。

更新 Tools for Windows PowerShell

執行 `Get-AWSPowerShellVersion` Cmdlet 來確認您正在執行的版本，並將此版本與 [PowerShell Gallery](#) 網站上可用的 Tools for Windows PowerShell 版本進行比較。我們建議您每兩到三週檢查一次。只有在您更新到具有該支援的版本之後，才能支援新的命令 AWS 和服務。

- 若您使用 `Install-Module` Cmdlet 進行安裝，請執行下列命令。

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

- 若您使用下載的 ZIP 檔案安裝：
 1. 從 [Tools for PowerShell](#) 網站下載最新版本。將所下載檔案名稱中的套件版本號碼，與您在執行 `Get-AWSPowerShellVersion` Cmdlet 時取得的版本號碼進行比較。
 2. 若下載版本的數字比您安裝的版本更高，請關閉所有 Tools for Windows PowerShell 主控台。
 3. 安裝較新版本 Tools for Windows PowerShell。

安裝完成後，請執行 `Import-Module AWSPowerShell` 來將更新後的 Cmdlet 載入您的 PowerShell 工作階段。或者，從開始功能表執行自訂 AWS Tools for PowerShell 主控台。

在 Linux 或 macOS AWS Tools for PowerShell 上安裝

本主題說明如何在 Linux 或 macOS AWS Tools for PowerShell 上安裝。

設定概觀

若要 AWS Tools for PowerShell 在 Linux 或 macOS 電腦上安裝，您可以從兩個套件選項中選擇：

- [AWS.Tools](#) – 的模組化版本 AWS Tools for PowerShell。每個 AWS 服務都由其個別的小型模組支援，具有共用支援模組 `AWS.Tools.Common`。
- [AWSPowerShell.NetCore](#) – 的單一大型模組版本 AWS Tools for PowerShell。此單一大型模組支援所有 AWS 服務。

Note

請注意，單一模組可能太大，無法與 [AWS Lambda](#) 功能一起使用。請改用上面的模組化版本。

在執行 Linux 或 macOS 的電腦上設定任何一個都會涉及下列任務，如本主題中稍後部分所說明：

1. 在支援的系統上安裝 PowerShell Core 6.0 或更新版本。
2. 在安裝 PowerShell Core 之後，請在您的系統 shell 中執行 pwsh 來啟動 PowerShell。
3. 安裝 AWS.Tools 或 AWSPowerShell.NetCore。
4. 執行適當的 Import-Module Cmdlet 來將模組匯入您的 PowerShell 工作階段。
5. 執行 [Initialize-AWSDefaultConfiguration](#) cmdlet 以提供您的 AWS 登入資料。

先決條件

若要執行 AWS Tools for PowerShell Core，您的電腦必須執行 PowerShell Core 6.0 或更新版本。

- 如需支援 Linux 平台的版本清單，以及如何在 Linux 類型電腦上安裝 PowerShell 最新版本的資訊，請參閱 Microsoft 網站上的[在 Linux 上安裝 PowerShell](#)。有些 Linux 類型作業系統 (例如 Arch、Kali 和 Raspbian) 不受支援，但擁有不同程度的社群支援。
- 如需支援 macOS 的版本，以及如何在 macOS 上安裝 PowerShell 的資訊，請參閱 Microsoft 網站上的[在 macOS 上安裝 PowerShell](#)。

在 Linux 或 macOS 上安裝 AWS.Tools

您可以在執行 PowerShell Core 6.0 或更新版本的電腦 AWS Tools for PowerShell 上安裝的模組化版本。如需如何安裝 PowerShell Core 的相關資訊，請參閱 Microsoft PowerShell 網站中的《[安裝各種版本的 PowerShell](#)》一文。

有三種方式可以安裝 AWS.Tools：

- 使用 AWS.Tools.Installer 模組中的 Cmdlet。此模組可簡化其他 AWS.Tools 模組的安裝和更新。AWS.Tools.Installer 需要 PowerShellGet，並會自動下載和安裝其更新版本。AWS.Tools.Installer 會自動讓您的模組版本保持同步。當您安裝或更新至某個模組的較新版本時，中的 cmdlet AWS.Tools.Installer 會自動將所有其他 AWS.Tools 模組更新為相同的版本。

此方法會在以下程序中說明。

- 從 [AWS.Tools.zip](#) 下載模組，然後解壓縮至其中一個模組目錄。您可以列印 `$Env:PSModulePath` 變數的值來探索您的模組目錄。
- 使用 Install-Module cmdlet 從 PowerShell Gallery 安裝每個服務模組。

使用 **AWS.Tools.Installer** 模組在 Linux 或 macOS **AWS.Tools** 上安裝

1. 執行以下命令來啟動 PowerShell Core 工作階段。

```
$ pwsh
```

Note

除非手邊任務需要，否則建議「請勿」以提高權限的管理員身分執行 PowerShell。這是因為可能會有潛在的安全風險，且不符合最低權限原則。

2. 若要使用 **AWS.Tools.Installer** 安裝模組化 **AWS.Tools** 套件，請執行以下命令。

```
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
```

若您收到通知，顯示儲存庫「不受信任」，表示系統正在詢問您是否無論如何都要進行安裝。輸入 **y** 來允許 PowerShell 安裝模組。若要在不信任儲存庫的情況下避免出現提示及安裝模組，您可以執行以下命令。

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

3. 您現在可以安裝每個您希望使用服務的模組。例如，以下命令會安裝 Amazon EC2 和 Amazon S3 模組。此命令也會安裝指定模組運作所需的任何相依模組。例如，當您安裝第一個 **AWS.Tools** 服務模組時，它也會安裝 **AWS.Tools.Common**。這是所有 AWS 服務模組所需的共用模組。它也會移除舊版模組，並將其他模組更新至相同的更新版本。

```
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

Install-AWSToolsModule Cmdlet 會從名為 PSGallery 的 PSRepository (<https://www.powershellgallery.com/>) 下載所有要求的模組，並將儲存庫視為信任的來源。如需此 PSRepository 的詳細資訊，請使用 Get-PSRepository -Name PSGallery 命令。

上一個命令會將模組安裝到系統上的預設目錄中。實際的目錄取決於您的作業系統發行版本和版本，以及您安裝的 PowerShell 版本。例如，如果您在類似 RHEL 的系統上安裝 PowerShell 7，則預設模組很可能位於 /opt/microsoft/powershell/7/Modules (或 \$PSHOME/Modules)，而使用者模組很可能位於 ~/.local/share/powershell/Modules。如需詳細資訊，請參閱 Microsoft PowerShell 網站上的[在 Linux 上安裝 PowerShell](#)。若要查看安裝模組的位置，請執行下列命令：

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

若要安裝其他模組，請使用適當的模組名稱執行類似的命令，例如 [PowerShell Gallery](#) 中的命令。

在 Linux 或 macOS 上安裝 AWSPowerShell.NetCore

若要升級到 AWSPowerShell.NetCore 的較新版本，請遵循 [在 Linux 或 macOS AWS Tools for PowerShell 上更新](#) 中的說明。請先解除安裝舊版 AWSPowerShell.NetCore。

您可以透過下列兩種方式安裝 `AWSPowerShell.NetCore`：

- 從 [AWSPowerShell.NetCore.zip](#) 下載模組，然後解壓縮至其中一個模組目錄。您可以列印 `$Env:PSModulePath` 變數的值來探索您的模組目錄。
- 使用 `Install-Module Cmdlet` 從 PowerShell Gallery 安裝，如下列程序中所描述。

使用 `Install-Module cmdlet` 在 Linux 或 macOS 上安裝 `AWSPowerShell.NetCore`

執行以下命令來啟動 PowerShell Core 工作階段。

```
$ pwsh
```

Note

建議「請勿」執行 `sudo pwsh` 來以提升的管理員權限啟動 PowerShell。這是因為可能會有潛在的安全風險，且不符合最低權限原則。

若要從 PowerShell Gallery 安裝 `AWSPowerShell.NetCore` 單一模組套件，請執行以下命令。

```
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
```

若您收到通知，顯示儲存庫「不受信任」，表示系統正在詢問您是否無論如何都要進行安裝。輸入 **y** 來允許 PowerShell 安裝模組。若要在不信任儲存庫的情況下避免出現提示，您可以執行以下命令。

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

除非您想要 AWS Tools for PowerShell 為電腦的所有使用者安裝，否則您不需要將此命令做為根目錄執行。若要執行此作業，請在您以 `sudo pwsh` 啟動的 PowerShell 工作階段中執行以下命令。

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

指令碼執行

Set-ExecutionPolicy 命令不適用於非 Windows 系統。您可以執行 Get-ExecutionPolicy，這會顯示在非 Windows 系統上執行的 PowerShell Core 的預設執行政策設定是 Unrestricted。如需詳細資訊，請參閱 Microsoft Technet 網站上的 [About Execution Policies](#)。

由於 PSModulePath 包含 AWS 模組目錄的位置，因此 Get-Module -ListAvailable cmdlet 會顯示您安裝的模組。

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	

設定 PowerShell 主控台以使用 AWS Tools for PowerShell Core (僅適用於 AWSPowerShell.NetCore)

PowerShell Core 通常會在您執行模組中的 Cmdlet 時自動載入模組。但這不適用於 AWSPowerShell.NetCore，因為其大小較大。若要開始執行 AWSPowerShell.NetCore Cmdlet，您必須先執行 Import-Module AWSPowerShell.NetCore 命令。AWS.Tools 模組中的 Cmdlet 不需要此操作。

初始化您的 PowerShell 工作階段

當您在安裝之後，在 Linux 型或 macOS 型系統上啟動 PowerShell 時 AWS Tools for PowerShell，您必須執行 [Initialize-AWSDefaultConfiguration](#)，以指定要使用的 AWS 存取金鑰。如需 Initialize-AWSDefaultConfiguration 的相關資訊，請參閱 [使用 AWS 登入資料](#)。

Note

在舊版 (3.3.96.0 之前) 中 AWS Tools for PowerShell，此 cmdlet 名為 Initialize-AWSDefaults。

版本控制

AWS AWS Tools for PowerShell 會定期發行 的新版本，以支援新的 AWS 服務和功能。若要判斷您已安裝的 AWS Tools for PowerShell 版本，請執行 [Get-AWSPowerShellVersion](#) cmdlet。

例如：

```
PS > Get-AWSPowerShellVersion
```

```
AWS Tools for PowerShell  
Version 4.1.849  
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
Amazon Web Services SDK for .NET  
Core Runtime Version 3.7.402.75  
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
Release notes: https://github.com/aws/aws-tools-for-powershell/blob/v4.1/changelogs/  
CHANGELOG.ALL.md
```

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

若要查看目前版本工具中支援的 AWS 服務清單，請將 -ListServiceVersionInfo 參數新增至 [Get-AWSPowerShellVersion](#) cmdlet。

若要確認您執行的 PowerShell 版本，請輸入 \$PSVersionTable 來檢視 \$PSVersionTable [自動變數](#)的內容。

例如：

```
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
```

在 Linux 或 macOS AWS Tools for PowerShell 上更新

定期 AWS Tools for PowerShell 發行更新版本的 時，您應該更新在本機執行的版本。

更新模組化 **AWS.Tools** 模組

若要將 AWS.Tools 模組更新至最新版本，請執行下列命令：

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

此命令會更新所有目前安裝的 AWS.Tools 模組，而對於已成功更新的模組，則會移除舊版。

Note

Update-AWSToolsModule Cmdlet 會從名為 PSGallery 的 PSRepository (<https://www.powershellgallery.com/>) 下載所有模組，並視為是信任的來源。如需此 PSRepository 的詳細資訊，請使用 Get-PSRepository -Name PSGallery 命令。

更新 Tools for PowerShell Core

執行 Get-AWSPowerShellVersion Cmdlet 來確認您正在執行的版本，並將此版本與 [PowerShell Gallery](https://www.powershellgallery.com/) 網站上可用的 Tools for Windows PowerShell 版本進行比較。我們建議您每兩到三週檢查一次。只有在您更新到具有該支援的版本之後，才能支援新的命令 AWS 和服務。

安裝 AWSPowerShell.NetCore 的較新版本前，請解除安裝現有模組。請在您解除安裝現有套件前關閉任何開啟的 PowerShell 工作階段。執行下列命令以解除安裝套件。

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

解除安裝套件後，請執行以下命令來安裝更新後的模組。

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

安裝完成後，請執行 Import-Module AWSPowerShell.NetCore 命令來將更新後的 Cmdlet 載入您的 PowerShell 工作階段。

相關資訊

- [開始使用 AWS Tools for Windows PowerShell](#)
- [在中使用 AWS 服務 AWS Tools for PowerShell](#)

從 3.3 AWS Tools for PowerShell 版遷移至 第 4 版

AWS Tools for PowerShell 第 4 版是 3.3 AWS Tools for PowerShell 版的回溯相容更新。此版本新增了大量改善的項目，同時保持現有的 Cmdlet 行為。

在升級至新版本後，現有指令碼應該能繼續運作，但建議您在升級生產環境前先進行徹底的測試。

本節說明這些變更並解說這些變更可能會對指令碼有何影響。

新全面模組化 AWS.Tools 版本

AWSPowerShell.NetCore 和 AWSPowerShell 套件為「整合式」。這表示在同一個模組中支援所有 AWS 服務，使其非常龐大，並且隨著新增每個新 AWS 服務和功能而變大。新 AWS.Tools 套件分為較小的模組，可讓您靈活地僅下載和安裝您使用 AWS 的服務所需的套件。此套件包含一個所有其他模組需要的共用 AWS.Tools.Common 模組，以及一個可視需要簡化模組安裝、更新及移除作業的 AWS.Tools.Installer 模組。

這也會在第一次呼叫時啟用 Cmdlet 的自動匯入功能，而不必先呼叫 Import-module。不過，若要在呼叫 Cmdlet 前與相關聯的 .NET 物件進行互動，您仍然必須呼叫 Import-Module 以讓 PowerShell 了解相關的 .NET 類型。

例如，以下命令句有 `Amazon.EC2.Model.Filter` 的參照。這種類型的參照無法觸發自動匯入功能，因此您必須先呼叫 `Import-Module`，否則命令則會失敗。

```
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
```

全新 `Get-AWSService` Cmdlet

為了協助您探索模組 `AWS.Tools` 集合中每個 AWS 服務的模組名稱，您可以使用 `Get-AWSService` cmdlet。

```
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
...
```

控制 Cmdlet 傳回之物件的全新 `-Select` 參數

第 4 版中的大多數 Cmdlet 可支援全新的 `-Select` 參數。每個 Cmdlet 都會使用適用於 .NET 的 AWS SDK 呼叫 AWS 服務 API。然後，AWS Tools for PowerShell 用戶端會將回應轉換為可在 PowerShell 指令碼和管道中使用的物件，並轉換為其他命令。有時候最終 PowerShell 物件在原始回應中會有比您所需更多的欄位或屬性，有時候您可能會希望物件在回應中包含預設不存在的欄位或屬性。`-Select` 參數可讓您指定 Cmdlet 所傳回之 .NET 物件中要包含的項目。

例如，[Get-S3Object](#) Cmdlet 會叫用 Amazon S3 開發套件操作 [ListObjects](#)。該操作會傳回 [ListObjectsResponse](#) 物件。不過，根據預設，Get-S3Object Cmdlet 只會將開發套件回應的 S3Objects 元素傳回給 PowerShell 使用者。在以下範例中，該物件是具有兩個元素的陣列。

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

```
ETag : "01234567890123456789012345678901111"
```

```
BucketName : amzn-s3-demo-bucket
```

```
Key : file1.txt
```

```
LastModified : 9/30/2019 1:31:40 PM
```

```
Owner : Amazon.S3.Model.Owner
```

```
Size : 568
```

```
StorageClass : STANDARD
```

```
ETag : "01234567890123456789012345678902222"
```

```
BucketName : amzn-s3-demo-bucket
```

```
Key : file2.txt
```

```
LastModified : 7/15/2019 9:36:54 AM
```

```
Owner : Amazon.S3.Model.Owner
```

```
Size : 392
```

```
StorageClass : STANDARD
```

在第 4 AWS Tools for PowerShell 版中，您可以指定 `-Select *` 傳回 SDK API 呼叫傳回的完整 .NET 回應物件。

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

```
IsTruncated      : False
```

```
NextMarker       :
```

```
S3Objects        : {file1.txt, file2.txt}
```

```
Name             : amzn-s3-demo-bucket
```

```
Prefix           :
```

```
MaxKeys          : 1000
```

```
CommonPrefixes   : {}
```

```
Delimiter        :
```

您也可以指定您想要的特定巢狀屬性的路徑。以下範例只會傳回 S3Objects 陣列中每個元素的 Key 屬性。

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

```
file1.txt
```

```
file2.txt
```

在某些情況下，傳回 Cmdlet 參數可能非常有用。您可以使用 `-Select ^ParameterName` 來達成此操作。此功能取代了 `-PassThru` 參數，該參數仍可取得但已遭取代。

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

每個 Cmdlet 的 [參考主題](#) 會識別是否支援 `-Select` 參數。

以更一致的方式限制輸出的項目數

舊版 AWS Tools for PowerShell 可讓您使用 `-MaxItems` 參數來指定最終輸出中傳回的物件數量上限。

AWS.Tools 已移除此行為。

在 AWSPowerShell.NetCore 和 AWSPowerShell 中，此行為已遭取代，而未來版本也會移除此行為。

如果基礎服務 API 支援 `MaxItems` 參數，則仍可依 API 指定取得及運作，但不會再有限制 Cmdlet 輸出中傳回之項目數的新增行為。

若要限制最終輸出中傳回的項目數，請將輸出輸送至 `Select-Object` Cmdlet 並指定 `-First n` 參數，其中 *n* 是要在最終輸出中包含的項目數上限。

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

並非所有 AWS 服務 `-MaxItems` 都以相同方式支援，因此這會移除該不一致的情況，以及有時發生的非預期結果。此外結合新 [-Select](#) 參數的 `-MaxItems` 有時可能會導致令人混淆的結果。

更易於使用的串流參數

`Stream` 或 `byte[]` 類型的參數現在可以接受 `string`、`string[]` 或 `FileInfo` 值。

例如，您可以使用以下任一範例。

```
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 `byte[]` 會使用 UTF-8 編碼將所有字串轉換為。

依屬性名稱擴充管道

為了提升使用者體驗的一致性，您現在可以指定「任何」參數的屬性名稱，藉此傳遞管道輸入。

在以下範例中，我們建立了一個自訂物件，其屬性具有符合目標 Cmdlet 參數名稱的名稱。當 Cmdlet 執行時，它會自動使用這些屬性做為其參數。

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

Note

有些屬性在舊版中支援此屬性 AWS Tools for PowerShell。第 4 版可讓您指定「所有」參數，讓此體驗更加一致。

靜態常見參數

為了提高 4.0 版的一致性 AWS Tools for PowerShell，所有參數都是靜態的。

在舊版中 AWS Tools for PowerShell，一些常見的參數，例如 `AccessKeySecretKey`、`ProfileName`、或 `Region` 是動態的，而所有其他參數都是靜態的。這可能會發生問題，因為 PowerShell 會先繫結靜態參數，然後再繫結動態參數。例如，假設您執行以下命令。

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

舊版 PowerShell 會將 `us-west-2` 值繫結至 `-RegionName` 靜態參數，而不是 `-Region` 動態參數。這可能會讓使用者產生混淆。

AWS.Tools 會宣告並強制執行必要參數

AWS.Tools.* 模組現在會宣告並強制執行必要的 Cmdlet 參數。當 AWS 服務宣告某個 API 參數為必要時，如果您尚未指定該參數，PowerShell 則會提示您對應的 Cmdlet 參數。這僅適用於 AWS.Tools。為確保回溯相容性，這不適用於 AWSPowerShell.NetCore 或 AWSPowerShell。

所有參數皆可為 Null

您現在可以將 \$null 指派給值類型參數 (數字和日期)。此變更應該不會影響現有的指令碼。這可讓您略過必要參數的提示。必要參數只會在 AWS.Tools 中強制執行。

如果您使用第 4 版執行以下範例，則會有效略過用戶端驗證，因為您會為每個必要參數提供「值」。不過，Amazon EC2 API 服務呼叫會失敗，因為 AWS 服務仍然需要該資訊。

```
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
```

移除先前取代的功能

下列功能已在舊版 中棄用 AWS Tools for PowerShell，並已在版本 4 中移除：

- 移除 Stop-EC2Instance Cmdlet 中的 -Terminate 參數。請改用 Remove-EC2Instance。
- 移除 Clear-AWSCredential Cmdlet 中的 -ProfileName 參數。請改用 Remove-AWSCredentialProfile。
- 移除 Import-EC2Instance 和 Import-EC2Volume Cmdlet。

開始使用 AWS Tools for Windows PowerShell

本節中的部分主題說明[建立工具](#)之後，使用 Tools for Windows PowerShell 的基本原理。例如，這些主題會說明如何指定 Tools for Windows PowerShell 與 AWS 互動時應使用的[憑證](#)和 [AWS 區域](#)。

本節中的其他主題提供了一些進階方法的資訊，您可以使用這些方法來設定工具、環境和專案。

主題

- [使用 設定工具身分驗證 AWS](#)
- [指定 AWS 區域](#)
- [使用 設定聯合身分 AWS Tools for PowerShell](#)
- [Cmdlet 探索和別名](#)
- [中的管道、輸出和反覆運算 AWS Tools for PowerShell](#)
- [憑證和設定檔解析](#)
- [有關使用者和角色的其他資訊](#)
- [使用舊版憑證](#)

使用 設定工具身分驗證 AWS

在使用 進行開發 AWS 時，您必須建立程式碼向 進行身分驗證的方式 AWS 服務。視環境和您可用的存取權而定，您可以透過不同方式設定 AWS 資源的程式設計 AWS 存取。

如要查看 Tools for PowerShell 的各種身份驗證方法，請參閱 AWS SDK 和工具參考指南中的[身份驗證和存取](#)。

本主題假設新的使用者正在本機開發，其雇主尚未提供身分驗證方法，且將使用 AWS IAM Identity Center 取得臨時登入資料。如果您的環境不適用這些假設，則本主題中的有些資訊可能不適用您的環境，或者有些資訊可能已經提供給您。

設定此環境需要幾個步驟，總結如下：

1. [啟用和設定 IAM Identity Center](#)
2. [設定 Tools for PowerShell 以使用 IAM Identity Center。](#)
3. [啟動 AWS 存取入口網站工作階段](#)

啟用和設定 IAM Identity Center

若要使用 AWS IAM Identity Center，必須先啟用和設定它。要查看有關如何在 PowerShell 執行此操作的詳細資訊，請參閱 AWS SDK 和工具參考指南中 [IAM Identity Center 身份驗證](#) 主題中的步驟 1。具體來說，請遵循我沒有透過 IAM Identity Center 建立存取權限下的任何必要說明。

設定 Tools for PowerShell 以使用 IAM Identity Center。

Note

從 Tools for PowerShell 4.1.538 版開始，設定 SSO 憑證和啟動 AWS 存取入口網站工作階段的建議方法是使用 [Initialize-AWSSSOConfiguration](#) 和 [Invoke-AWSSSOLogin](#) cmdlet，如本主題所述。如果您無法存取該版本的 Tools for PowerShell（或更新版本）或無法使用這些 cmdlet，您仍然可以使用執行這些任務 AWS CLI。若要了解如何操作，請參閱 [使用 AWS CLI 進行入口網站登入](#)。

下列程序會使用 Tools for PowerShell 用來取得臨時登入資料的 SSO 資訊來更新共用 AWS config 檔案。由於此程序，也會啟動 AWS 存取入口網站工作階段。如果共用 config 檔案已有 SSO 資訊，而您只想了解如何使用 Tools for PowerShell 啟動存取入口網站工作階段，請參閱本主題的下一節：[啟動 AWS 存取入口網站工作階段](#)。

1. 如果您尚未這麼做，請開啟 PowerShell 並 AWS Tools for PowerShell 安裝適合您作業系統和環境的，包括常見的 cmdlet。如需如何進行該服務的詳細資訊，請參閱[安裝 AWS Tools for PowerShell](#)。

例如，如果在 Windows 上安裝 Tools for PowerShell 的模組化版本，您很可能執行類似以下的命令：

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

2. 執行下列命令。將範例屬性值取代為 IAM Identity Center 組態中的值。如需有關這些屬性以及如何尋找它們的資訊，請參閱 AWS SDKs 和工具參考指南中的 [IAM Identity Center 憑證提供者設定](#)。

```
$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
```

或者，您可以直接使用 cmdlet `Initialize-AWSSSOConfiguration` 和 Tools for PowerShell 會提示您輸入屬性值。

特定屬性值的考量事項：

- 如果您只按照指示[啟用和設定 IAM Identity Center](#)，則 `RoleName` 可能是 `PowerUserAccess`。但是，如果您特別為 PowerShell 工作建立 IAM Identity Center 許可集，請改用它。
 - 請務必使用您已設定 IAM Identity Center AWS 區域的。
3. 此時，共用 AWS config 檔案包含名為 `my-sso-profile` 的設定檔，其中包含一組組態值，可從 Tools for PowerShell 參考。若要尋找此檔案的位置，請參閱 AWS SDK 和工具參考指南中的[共用檔案位置](#)。

Tools for PowerShell 會使用設定檔的 SSO 權杖提供者，在傳送要求至 AWS 之前取得憑證。此 `sso_role_name` 值是連接至 IAM Identity Center 許可集的 IAM 角色，應允許存取您應用程式中 AWS 服務使用的。

下列範例顯示使用上述命令建立的設定檔。某些屬性值及其順序在您的實際設定檔中可能不同。設定檔的 `sso-session` 屬性是指名為 `my-sso-session` 的區段，其中包含啟動 AWS 存取入口網站工作階段的設定。

```
[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. 如果您已有作用中的 AWS 存取入口網站工作階段，Tools for PowerShell 會通知您已登入。

如果不是這種情況，Tools for PowerShell 會嘗試在您的預設 Web 瀏覽器中自動開啟 SSO 授權頁面。請遵循瀏覽器中的提示，其中可能包含 SSO 授權碼、使用者名稱和密碼，以及存取 AWS IAM Identity Center 帳戶和許可集的許可。

適用於 PowerShell 的工具會通知您 SSO 登入成功。

啟動 AWS 存取入口網站工作階段

在執行存取的命令之前 AWS 服務，您需要作用中的 AWS 存取入口網站工作階段，以便 Tools for PowerShell 可以使用 IAM Identity Center 身分驗證來解析登入資料。若要登入 AWS 存取入口網站，請在 PowerShell 中執行下列命令，其中 `-ProfileName my-sso-profile` 是當您遵循本主題上一節中的程序時，在共用 config 檔案中建立的設定檔名稱。

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

如果您已有作用中的 AWS 存取入口網站工作階段，Tools for PowerShell 會通知您已登入。

如果不是這種情況，Tools for PowerShell 會嘗試在您的預設 Web 瀏覽器中自動開啟 SSO 授權頁面。請遵循瀏覽器中的提示，其中可能包含 SSO 授權碼、使用者名稱和密碼，以及存取 AWS IAM Identity Center 帳戶和許可集的許可。

適用於 PowerShell 的工具會通知您 SSO 登入成功。

若要測試您是否已經有作用中的工作階段，請在安裝或匯入 `AWS.Tools.SecurityToken` 模組後視需要執行下列命令。

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

對 `Get-STSCallerIdentity` cmdlet 的回應會報告共用 config 檔案中設定的 IAM Identity Center 帳戶和許可集。

範例

以下是如何將 IAM Identity Center 與 Tools for PowerShell 搭配使用的範例。假設如下：

- 您已啟用 IAM Identity Center，並依照本主題先前所述進行設定。SSO 屬性位於設定檔中，此 `my-sso-profile` 設定檔先前在本主題中已設定。

- 當您透過 `Initialize-AWSSSOConfiguration` 或 `Invoke-AWSSSOLogin` cmdlet 登入時，使用者至少擁有 Amazon S3 的唯讀許可。
- 部分 S3 儲存貯體可供該使用者檢視。

視需要安裝或匯入 `AWS.Tools.S3` 模組，然後使用下列 PowerShell 命令來顯示 S3 儲存貯體的清單。

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

其他資訊

- 有關 Tools for PowerShell 驗證的更多選項，例如使用設定檔和環境變量，請參閱 AWS SDK 和工具參考指南中的[組態](#)章節。
- 有些命令需要指定 AWS 區域。有多種方法可以執行此操作，包括 `-Region` cmdlet 選項、`[default]` 設定檔和 `AWS_REGION` 環境變數。如需詳細資訊，請參閱本指南[指定 AWS 區域](#)中的和 AWS SDKs 和工具參考指南中的[AWS 區域](#)。
- 如需了解有關最佳實務的資訊，請參閱 IAM 使用者指南中的[IAM 安全最佳實務](#)。
- 若要建立短期 AWS 登入資料，請參閱《IAM 使用者指南》中的[暫時安全登入](#)資料。
- 若要瞭解其他憑證提供者，請參閱 AWS SDK 和工具參考指南中的[標準化憑證提供者](#)。

主題

- [使用 AWS CLI 進行入口網站登入](#)

使用 AWS CLI 進行入口網站登入

從 Tools for PowerShell 4.1.538 版開始，設定 SSO 憑證和啟動 AWS 存取入口網站工作階段的建議方法是使用 [Initialize-AWSSSOConfiguration](#) 和 [Invoke-AWSSSOLogin](#) cmdlet，如中所述[使用設定工具身分驗證 AWS](#)。如果您無法存取該版本的 Tools for PowerShell（或更新版本）或無法使用這些 cmdlet，您仍然可以使用執行這些任務 AWS CLI。

設定 Tools for PowerShell 透過使用 IAM Identity Center AWS CLI。

如果您尚未這麼做，請務必在繼續之前[啟用和設定 IAM Identity Center](#)。

有關如何透過設定 Tools for PowerShell 以使用 IAM Identity Center 的資訊 AWS CLI，請參閱 AWS SDKs 和 Tools 參考指南中的[IAM Identity Center 身分驗證](#)主題中的步驟 2。完成此組態之後，您的系統應該包含下列元素：

- 在執行應用程式之前 AWS CLI，您用來啟動 AWS 存取入口網站工作階段的。
- 包含 [\[default\] 設定檔](#) 的共用 AWS config 檔案，其中包含一組組態值，可從 Tools for PowerShell 參考。若要尋找此檔案的位置，請參閱 AWS SDK 和工具參考指南中的 [共用檔案位置](#)。Tools for PowerShell 會使用設定檔的 SSO 權杖提供者，在傳送要求至 AWS 之前取得憑證。此 `sso_role_name` 值是連接至 IAM Identity Center 許可集的 IAM 角色，應允許存取您應用程式中 AWS 服務使用的。

下列範例 config 檔案顯示使用 SSO 字符提供者設定的 `[default]` 設定檔。設定檔的 `sso_session` 設定是指已命名的 `sso-session` 區段。`sso-session` 區段包含啟動 AWS 存取入口網站工作階段的設定。

```
[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

您的 PowerShell 工作階段必須安裝並匯入下列模組，以便 SSO 解析能夠運作：

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

如果您使用的是舊版的 Tools for PowerShell，但沒有這些模組，您會收到類似下列的錯誤：
「找不到組件 AWSSDK.SSOIDC...」。

啟動 AWS 存取入口網站工作階段

在執行存取的命令之前 AWS 服務，您需要作用中的 AWS 存取入口網站工作階段，以便 Tools for Windows PowerShell 可以使用 IAM Identity Center 身分驗證來解析登入資料。視您設定的工作階段長

度而定，您的存取最終會過期，而 Tools for Windows PowerShell 會遇到驗證錯誤。若要登入 AWS 存取入口網站，請在 中執行下列命令 AWS CLI。

```
aws sso login
```

由於您使用的是 [default] 設定檔，因此您不需要使用 `--profile` 選項呼叫 命令。如果您的 SSO 權杖提供者組態使用具名設定檔，則命令會 `aws sso login --profile named-profile` 改為 。如需具名設定檔的詳細資訊，請參閱 AWS SDKs 和工具參考指南中的 [設定檔](#) 一節。

若要測試您是否已經有作用中的工作階段，請執行下列 AWS CLI 命令（具有與具名設定檔相同的考量）：

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

對此命令的回應，應報告共用 config 檔案中設定的 IAM Identity Center 帳戶和許可集合。

Note

如果您已有作用中的 AWS 存取入口網站工作階段並執行 `aws sso login`，則不需要提供登入資料。

登入程序可能會提示您允許 AWS CLI 存取您的資料。由於 AWS CLI 建置在適用於 Python 的 SDK 之上，因此許可訊息可能包含 `botocore` 名稱的變化。

範例

以下是如何將 IAM Identity Center 與 Tools for PowerShell 搭配使用的範例。假設如下：

- 您已啟用 IAM Identity Center，並依照本主題先前所述進行設定。SSO 屬性位於 [default] 設定檔中。
- 當您 AWS CLI 使用 透過 登入時 `aws sso login`，該使用者至少具有 Amazon S3 的唯讀許可。
- 部分 S3 儲存貯體可供該使用者檢視。

使用下列 PowerShell 命令來顯示 S3 儲存貯體的清單：

```
Install-Module AWS.Tools.Installer  
Install-AWSToolsModule S3
```

```
# And if using an older version of the AWS Tools for PowerShell:
Install-AWSToolsModule SS0, 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.SS0
Import-Module AWS.Tools.SS00IDC

# Older versions of the AWS Tools for PowerShell don't support the SS0 login flow, so
login with the CLI
aws sso login

# Now we can invoke cmdlets using the SS0 profile
Get-S3Bucket
```

如上所述，由於您正在使用 [default] 設定檔，因此您不需要使用 `-ProfileName` 選項呼叫 `Get-S3Bucket` cmdlet。如果您的 SSO 權杖提供者組態使用已命名的設定檔，則命令為 `Get-S3Bucket -ProfileName named-profile`。如需具名設定檔的詳細資訊，請參閱 AWS SDKs 和工具參考指南中的[設定檔](#)一節。

其他資訊

- 有關 Tools for PowerShell 驗證的更多選項，例如使用設定檔和環境變量，請參閱 AWS SDK 和工具參考指南中的[組態](#)章節。
- 有些命令需要指定 AWS 區域。有多種方法可以執行此操作，包括 `-Region` cmdlet 選項、[default] 設定檔和 `AWS_REGION` 環境變數。如需詳細資訊，請參閱本指南[指定 AWS 區域](#)中的和 AWS SDKs 和工具參考指南中的[AWS 區域](#)。
- 如需了解有關最佳實務的資訊，請參閱 IAM 使用者指南中的[IAM 安全最佳實務](#)。
- 若要建立短期 AWS 登入資料，請參閱《IAM 使用者指南》中的[暫時安全登入](#)資料。
- 若要瞭解其他憑證提供者，請參閱 AWS SDK 和工具參考指南中的[標準化憑證提供者](#)。

指定 AWS 區域

有兩種方式可在執行 AWS Tools for PowerShell 命令時指定要使用 AWS 的區域：

- 在個別命令上使用 `-Region` 通用參數。
- 使用 `Set-DefaultAWSRegion` 命令來設定所有命令的預設區域。

如果 Tools for Windows PowerShell 無法找出要使用的區域，許多 AWS cmdlet 會失敗。例外狀況包括 [Amazon S3](#)、Amazon SES 和 的 cmdlet AWS Identity and Access Management，其會自動預設為全域端點。

指定單一 AWS 命令的區域

將 `-Region` 參數新增到您的命令，如下。

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

若要為目前工作階段中的所有 AWS CLI 命令設定預設區域

在 PowerShell 命令提示字元中，輸入下列命令。

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

Note

此設定僅會存在於目前的工作階段。請依照您在 `Import-Module` 命令中的作法，將此命令新增至 PowerShell 設定檔，即可將該設定套用至所有 PowerShell 工作階段。

檢視所有 CLI AWS 命令的目前預設區域

在 PowerShell 命令提示字元中，輸入下列命令。

```
PS > Get-DefaultAWSRegion
```

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

清除所有 CLI AWS 命令的目前預設區域

在 PowerShell 命令提示字元中，輸入下列命令。

```
PS > Clear-DefaultAWSRegion
```

檢視所有可用 AWS 區域的清單

在 PowerShell 命令提示字元中，輸入下列命令。範例輸出的第三個欄位會識別出目前工作階段的預設區域。

```
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

系統可能會支援部分區域，但不會在 `Get-AWSRegion` Cmdlet 的輸出中包含該區域。例如，這有時對還不是全域的區域是如此。如果新增 `-Region` 參數至命令仍無法指定區域，請改為嘗試指定自訂端點中的區域，如下節所示。

指定自訂或非標準端點

請採用下列範例格式，將 `-EndpointUrl` 常用參數新增至 Tools for Windows PowerShell 命令，即可將自訂端點指定為 URL。

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

以下範例是採用 `Get-EC2Instance` cmdlet。在本範例中，自訂端點位於 `us-west-2`，亦稱美國西部 (奧勒岡)，但您可以使用任何其他支援的 AWS 區域 (包括 `Get-AWSRegion` 未列舉的區域)。

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

其他資訊

如需 AWS 區域的詳細資訊，請參閱 AWS SDKs 和工具參考指南中的 [AWS 區域](#)。

使用 設定聯合身分 AWS Tools for PowerShell

若要讓組織中的使用者存取 AWS 資源，您必須針對安全性、可稽核性、合規性，以及支援角色和帳戶區隔的功能，設定標準且可重複的身分驗證方法。雖然提供使用者存取 AWS APIs 的能力很常見，但沒有聯合 API 存取，您也必須建立 AWS Identity and Access Management (IAM) 使用者，這會破壞使用聯合的目的。本主題說明 中的 SAML (安全性聲明標記語言) 支援 AWS Tools for PowerShell，可簡化您的聯合存取解決方案。

中的 SAML 支援 AWS Tools for PowerShell 可讓您提供使用者聯合存取 AWS 服務。SAML 是以 XML 為基礎的開放標準格式，可在服務之間傳輸使用者身分驗證和授權資料；特別是身分提供者 (例如 [Active Directory Federation Services](#)) 與服務提供者 (例如) 之間 AWS。如需有關 SAML 及其運作方式的詳細資訊，請參閱 Wikipedia 上的 [SAML](#)，或「結構化資訊標準促進組織」(OASIS) 網站上的 [SAML 技術規格](#)。中的 SAML 支援與 SAML 2.0 AWS Tools for PowerShell 相容。

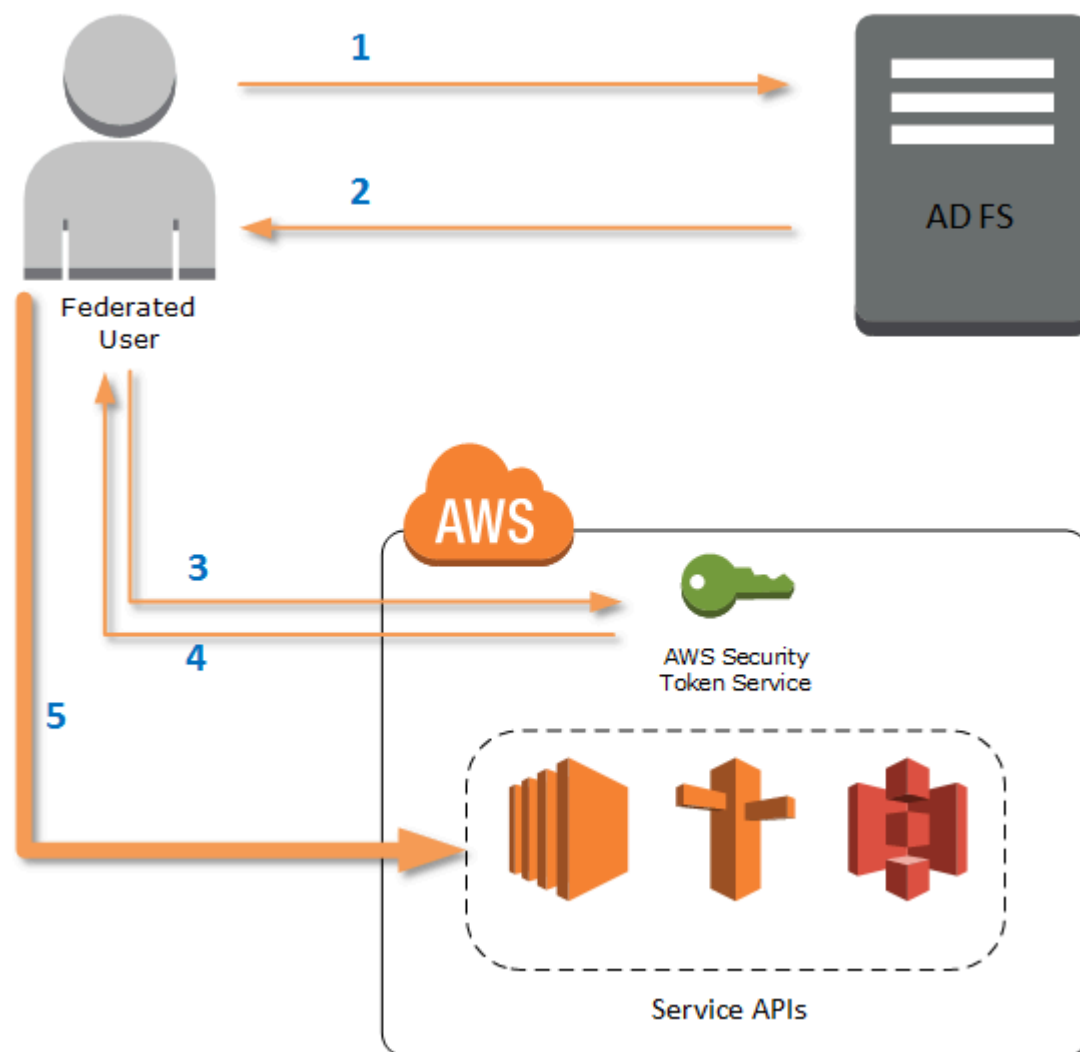
先決條件

您必須具備以下項目，才能首次嘗試使用 SAML 支援。

- 與您的 AWS 帳戶正確整合的聯合身分解決方案，可以僅使用您的組織登入資料進行主控台存取。如需如何專門針對 Active Directory Federation Services 執行此操作的詳細資訊，請參閱《IAM 使用者指南》中的[關於 SAML 2.0 聯合](#)，以及部落格文章：[AWS 使用 Windows Active Directory、AD FS 和 SAML 2.0 啟用聯合](#)。雖然部落格文章說明的是 AD FS 2.0，但 AD FS 3.0 的步驟也類似。
- 安裝在本機工作站 AWS Tools for PowerShell 上的 3.1.31.0 版或更新版本。

聯合身分使用者如何取得 AWS 服務 APIs 聯合存取權

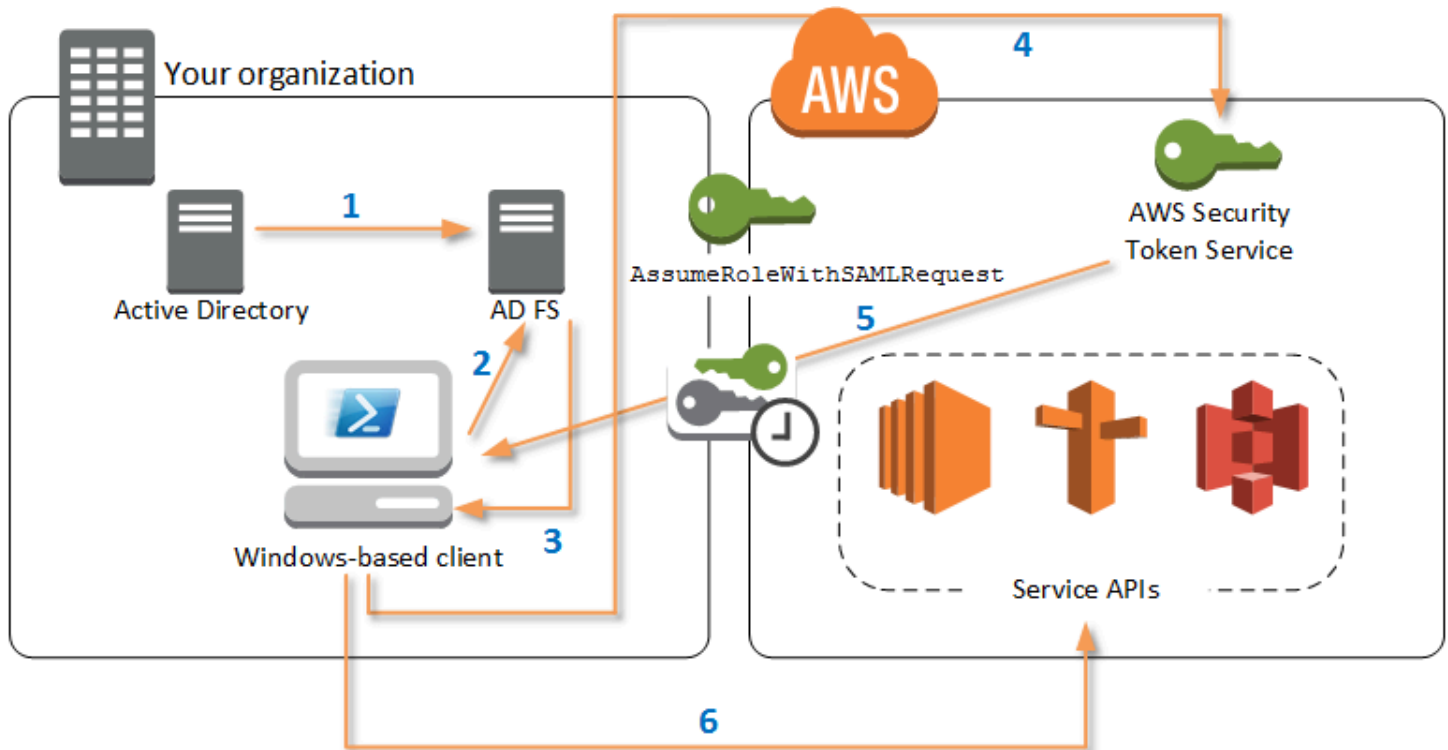
下列程序在高階描述 AD FS 如何聯合 Active Directory (AD) 使用者以取得 AWS 資源的存取權。



1. 聯合身分使用者電腦上的用戶端會驗證 AD FS。
2. 如果驗證成功，AD FS 會向使用者傳送 SAML 聲明。
3. 使用者的用戶端會將 SAML 聲明傳送至 AWS Security Token Service (STS)，做為 SAML 聯合請求的一部分。
4. STS 會傳回 SAML 回應，其中包含使用者可擔任之角色的 AWS 臨時登入資料。
5. 使用者透過在 提出的請求中包含這些臨時登入資料來存取 AWS 服務 APIs AWS Tools for PowerShell。

SAML 支援如何在 中運作 AWS Tools for PowerShell

本節說明 AWS Tools for PowerShell cmdlet 如何為使用者啟用 SAML 型聯合身分的組態。



1. AWS Tools for PowerShell 會使用 Windows 使用者目前的登入資料對 AD FS 進行身分驗證，或在使用者嘗試執行需要登入資料才能呼叫的 cmdlet 時，以互動方式進行身分驗證 AWS。
2. AD FS 驗證使用者。
3. AD FS 會產生包含聲明的 SAML 2.0 身分驗證回應；聲明的目的是識別並提供使用者的相關資訊。會從 SAML 聲明中 AWS Tools for PowerShell 擷取使用者授權角色的清單。
4. AWS Tools for PowerShell 透過發出 `AssumeRoleWithSAMLRequest` API 呼叫，將 SAML 請求，包括請求角色的 Amazon Resource Name (ARN) 轉送至 STS。
5. 如果 SAML 請求是有效的，STS 會傳回回應，其中包含 AWS、AccessKeyId、SecretAccessKey 和 SessionToken。這些登入資料可持續 3,600 秒 (1 小時)。
6. 使用者現在擁有有效的登入資料，可使用使用者角色有權存取的任何 AWS 服務 APIs。AWS Tools for PowerShell 會自動將這些登入資料套用至任何後續 AWS API 呼叫，並在過期時自動續約。

Note

當憑證過期且需要新的登入資料時，AWS Tools for PowerShell 會使用 AD FS 自動重新驗證，並取得下個小時的新登入資料。對於已加入網域的使用者帳戶，此程序會以無提示的方

式進行。對於未加入網域的帳戶，AWS Tools for PowerShell 會提示使用者輸入其登入資料，然後才能重新驗證。

如何使用 PowerShell SAML 組態 Cmdlet

AWS Tools for PowerShell 包含兩個提供 SAML 支援的新 cmdlet。

- `Set-AWSSamlEndpoint` 會設定您的 AD FS 端點、指派易記的名稱給端點，以及選擇性地說明端點的身分驗證類型。
- `Set-AWSSamlRoleProfile` 會建立或編輯您想使其與 AD FS 端點建立關聯的使用者帳戶描述檔，透過指定您提供給 `Set-AWSSamlEndpoint` cmdlet 的易記名稱來識別。每個角色描述檔對應到使用者獲權執行的單一角色。

如同 AWS 登入資料描述檔，您會為角色描述檔指派易記的名稱。您可以使用相同的易記名稱搭配 `Set-AWSCredential` cmdlet，或做為叫用 AWS 服務 APIs `-ProfileName` 參數值。

開啟新的 AWS Tools for PowerShell 工作階段。如果您執行的是 PowerShell 3.0 或更新版本，則當您執行其任一 cmdlet 時，都會自動匯入 AWS Tools for PowerShell 模組。如果您執行的是 PowerShell 2.0，則必須執行 `Import-Module` cmdlet，以手動方式匯入模組，如下列範例所示。

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

如何執行 `Set-AWSSamlEndpoint` 和 `Set-AWSSamlRoleProfile` Cmdlet

1. 首先，設定 AD FS 系統的端點設定。最簡單的方式是將端點存放在變數中，如本步驟所示。請務必將預留位置帳戶 ID 和 AD FS 主機名稱更換為您自己的帳戶 ID 和 AD FS 主機名稱。在 `Endpoint` 參數中指定 AD FS 主機名稱。

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

2. 若要建立端點設定，請執行 `Set-AWSSamlEndpoint` cmdlet，指定正確的 `AuthenticationType` 參數值。有效值包括 `Basic`、`Digest`、`Kerberos`、`Negotiate` 及 `NTLM`。如果您未指定此參數，預設值是 `Kerberos`。

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

Cmdlet 會使用 `-StoreAs` 參數傳回您指派的易記名稱，因此您可以在下一行中執行 `Set-AWSSamlRoleProfile` 時使用它。

- 現在，請執行 `Set-AWSSamlRoleProfile` cmdlet 以向 AD FS 身分提供者驗證身分，並取得讓使用者獲得授權能夠 (在 SAML 聲明中) 執行的一組角色。

`Set-AWSSamlRoleProfile` cmdlet 使用傳回的一組角色，來提示使用者選取角色以關聯至指定的描述檔，或驗證參數中提供的角色資料是否存在 (如果不存在，系統會提示使用者進行選擇)。如果使用者只獲得一個角色的授權，cmdlet 會自動將該角色關聯至描述檔，而不會提示使用者。不需要提供登入資料即可設定已加入網域的描述檔。

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

或者，對於 non-domain-joined 的帳戶，您可以提供 Active Directory 登入資料，然後選取使用者可存取 AWS 的角色，如下行所示。如果您有不同的 Active Directory 使用者帳戶，這可用來區分您組織內的角色 (例如，系統管理函數)。

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

- 在任何一種情況下，`Set-AWSSamlRoleProfile` cmdlet 都會提示您選擇應在描述檔中存放哪個角色。下列範例顯示兩個可用的角色：ADFS-Dev 和 ADFS-Production。IAM 角色與 AD FS 系統管理員的 AD 登入資料相關聯。

```
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"):
```

或者，您可以輸入 RoleARN、PrincipalARN 和選用的 NetworkCredential 參數，在沒有提示的情況下指定角色。如果身分驗證傳回的聲明中未列出指定的角色，系統會提示使用者從可用的角色中選擇。

```
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. 您可以在單一命令中新增 `StoreAllRoles` 參數，為所有角色建立描述檔，如以下程式碼所示。請注意，角色名稱將用於描述檔名稱。

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

如何使用角色設定檔來執行需要 AWS 登入資料的 Cmdlet

若要執行需要 AWS 登入資料的 cmdlet，您可以使用 AWS 共用登入資料檔案中定義的角色描述檔。將角色描述檔的名稱提供給 `Set-AWSCredential` (或作為中任何 `ProfileName` 參數的值 AWS Tools for PowerShell)，以自動取得描述檔中所述角色的臨時 AWS 登入資料。

雖然您一次只能使用一個角色描述檔，但可以在 shell 工作階段中切換描述檔。當您執行 `Set-AWSCredential` cmdlet 其本身時，此 cmdlet 不會驗證並取得登入資料；cmdlet 會記錄您想要使用指定的角色描述檔。除非您執行需要 AWS 登入資料的 cmdlet，否則無需身分驗證或請求登入資料。

您現在可以使用透過 `SAMLDemoProfile` 設定檔取得的臨時 AWS 登入資料來使用 AWS 服務 APIs。以下章節示範如何使用角色描述檔的範例。

範例 1：使用 `Set-AWSCredential` 設定預設角色

此範例使用 設定 AWS Tools for PowerShell 工作階段的預設角色 `Set-AWSCredential`。接著，您可以執行需要登入資料的 cmdlet，並獲得指定角色的授權。此範例列出美國西部 (奧勒岡) 區域的所有 Amazon Elastic Compute Cloud 執行個體，與您透過 `Set-AWSCredential` cmdlet 指定的描述檔相關聯。

```
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}

範例 2：在 PowerShell 工作階段期間變更角色描述檔

此範例列出與SAMLDemoProfile設定檔相關聯之角色 AWS 帳戶中所有可用的 Amazon S3 儲存貯體。此範例顯示，雖然您可能已在 AWS Tools for PowerShell 工作階段中稍早使用另一個描述檔，但您可以使用支援該描述檔的 cmdlet 為 -ProfileName 參數指定不同的值來變更描述檔。這對於從 PowerShell 命令列管理 Amazon S3 的管理員來說，是常見的任務。

```
PS > Get-S3Bucket -ProfileName SAMLDemoProfile
```

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

請注意，Get-S3Bucket cmdlet 指定透過執行 Set-AWSSamlRoleProfile cmdlet 而建立之描述檔的名稱。如果您稍早在工作階段中已設定角色描述檔 (例如透過執行 Set-AWSCredential cmdlet)，但想為 Get-S3Bucket cmdlet 使用不同的角色描述檔，這個命令會很有用。描述檔經理會讓臨時登入資料可用於 Get-S3Bucket cmdlet。

雖然登入資料在 1 小時 (STS 強制執行的限制) 後過期，但 AWS Tools for PowerShell 會在工具偵測到目前登入資料過期時請求新的 SAML 聲明，以自動重新整理登入資料。

對於已加入網域的使用者，此程序不會中斷，因為目前使用者的 Windows 身分已用於身分驗證。對於 non-domain-joined 使用者帳戶，AWS Tools for PowerShell 會顯示請求使用者密碼的 PowerShell 登入資料提示。使用者提供的登入資料，將用於重新驗證使用者並取得新的聲明。

範例 3：在區域中取得執行個體

以下範例列出亞太區域 (雪梨) 區域的所有 Amazon EC2 執行個體，與 ADFS-Production 描述檔使用的帳戶相關聯。這個實用命令很適合用來傳回一個區域中的所有 Amazon EC2 執行個體。


```
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

其他閱讀資料

如需如何實作聯合 API 存取的詳細資訊，請參閱[如何使用 SAML 2.0 為聯合 API/CLI 存取實作一般解決方案](#)。

如需支援問題或評論，請造訪適用於 [PowerShell 指令碼的](#) AWS 開發人員論壇或 [.NET Development](#)。

Cmdlet 探索和別名

本節說明如何列出 支援的服務 AWS Tools for PowerShell、如何顯示 提供的一組 Cmdlet AWS Tools for PowerShell 以支援這些服務，以及如何尋找替代 Cmdlet 名稱（也稱為別名）以存取這些服務。

Cmdlet 探索

所有 AWS 服務操作（或 APIs）都會記錄在每項服務的 API 參考指南中。舉例來說，請參閱 [IAM API 參考](#)。在大多數情況下，AWS 服務 API 和 AWS PowerShell Cmdlet one-to-one 通訊。若要取得與服務 AWS API 名稱對應的 cmdlet 名稱，請使用 `AWS -ApiOperation` 參數和服務 AWS API 名稱執行 `Get-AWSCmdletName` cmdlet。例如，若要取得以任何可用 `DescribeInstances` AWS 服務 API 為基礎的所有可能 cmdlet 名稱，請執行下列命令：

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

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

Get-GMLInstance	DescribeInstances	Amazon GameLift Service	GML

-ApiOperation 參數是預設參數，因此您可以省略參數名稱。以下範例相當於先前的範例：

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

如果您知道 API 和服務的名稱，您可以包含 -Service 參數以及 cmdlet 名詞字首或服務 AWS 名稱的一部分。例如，適用於 Amazon EC2 的 Cmdlet 名詞字首為 EC2。若要取得對應到 Amazon EC2 服務中 DescribeInstances API 的 Cmdlet 名稱，請執行以下其中一個命令。他們都會產生相同的輸出：

```
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

這些命令的參數值有區分大小寫。

如果您不知道所需 AWS 服務 API 或服務的名稱 AWS，您可以使用 -ApiOperation 參數，以及要比對的模式，以及 -MatchWithRegex 參數。例如，若要取得包含 SecurityGroup 的所有可用 Cmdlet 名稱，執行以下命令：

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

CmdletName	ServiceOperation
Approve-ECCacheSecurityGroupIngress	AuthorizeCacheSecurityGroupIngress
Get-ECCacheSecurityGroup	DescribeCacheSecurityGroups
New-ECCacheSecurityGroup	CreateCacheSecurityGroup
Remove-ECCacheSecurityGroup	DeleteCacheSecurityGroup
Revoke-ECCacheSecurityGroupIngress	RevokeCacheSecurityGroupIngress

Add-EC2SecurityGroupToClientVpnTargetNetwork			
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	

如果您知道 AWS 服務的名稱，但不知道 AWS 服務 API，請同時包含 `-MatchWithRegex` 參數和 `-Service` 參數，將搜尋範圍縮小為單一服務。例如，若只要取得 Amazon EC2 服務中內含 SecurityGroup 的所有 Cmdlet 名稱，請執行以下命令

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

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

如果您知道 AWS Command Line Interface (AWS CLI) 命令的名稱，您可以使用 `-AwsCliCommand` 參數和所需的 AWS CLI 命令名稱來取得以相同 API 為基礎的 cmdlet 名稱。例如，若要取得與 Amazon EC2 服務中 `authorize-security-group-ingress` AWS CLI 命令呼叫對應的 cmdlet 名稱，請執行下列命令：

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

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

`Get-AWSCmdletName` cmdlet 只需要足夠的 AWS CLI 命令名稱來識別服務和 AWS API。

若要取得 Tools for PowerShell Core 中所有 Cmdlet 的清單，請執行 PowerShell `Get-Command Cmdlet`，如以下範例所示。

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

您可以搭配 `-Module AWSPowerShell` 執行相同的命令，來查看 AWS Tools for Windows PowerShell 中的 Cmdlet。

`Get-Command Cmdlet` 產生的 Cmdlet 清單會以字母順序排序。請注意，根據預設，清單會根據 PowerShell 動詞排序，而非 PowerShell 名詞。

若要改為根據服務來排序結果，請執行以下命令：

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

若要篩選由 `Get-Command Cmdlet` 傳回的 Cmdlet，請將輸出輸送到 PowerShell `Select-String Cmdlet`。例如，若要檢視使用 AWS 區域的一組 cmdlet，請執行下列命令：

```
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

```

您亦可篩選 cmdlet 名詞的服務字首，藉此尋找特定服務的 cmdlet。若要查看可用服務字首的清單，請執行 `Get-AWSPowerShellVersion -ListServiceVersionInfo`。以下範例會傳回支援 Amazon CloudWatch Events 服務的 Cmdlet。

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

CommandType	Name	Version	Source
-----	----	-----	-----
Cmdlet	Add-CWResourceTag AWSPowerShell.NetCore	3.3.563.1	
Cmdlet	Disable-CWEventSource AWSPowerShell.NetCore	3.3.563.1	
Cmdlet	Disable-CWERule AWSPowerShell.NetCore	3.3.563.1	
Cmdlet	Enable-CWEventSource AWSPowerShell.NetCore	3.3.563.1	
Cmdlet	Enable-CWERule AWSPowerShell.NetCore	3.3.563.1	
Cmdlet	Get-CWEventBus AWSPowerShell.NetCore	3.3.563.1	
Cmdlet	Get-CWEventBusList AWSPowerShell.NetCore	3.3.563.1	
Cmdlet	Get-CWEventSource AWSPowerShell.NetCore	3.3.563.1	
Cmdlet	Get-CWEventSourceList AWSPowerShell.NetCore	3.3.563.1	
Cmdlet	Get-CWPartnerEventSource AWSPowerShell.NetCore	3.3.563.1	
Cmdlet	Get-CWPartnerEventSourceAccountList AWSPowerShell.NetCore	3.3.563.1	
Cmdlet	Get-CWPartnerEventSourceList AWSPowerShell.NetCore	3.3.563.1	
Cmdlet	Get-CWResourceTag AWSPowerShell.NetCore	3.3.563.1	
Cmdlet	Get-CWERule AWSPowerShell.NetCore	3.3.563.1	

Cmdlet	Get-CWERuleDetail	3.3.563.1
	AWSPowerShell.NetCore	
Cmdlet	Get-CWERuleNamesByTarget	3.3.563.1
	AWSPowerShell.NetCore	
Cmdlet	Get-CWETargetsByRule	3.3.563.1
	AWSPowerShell.NetCore	
Cmdlet	New-CWEEventBus	3.3.563.1
	AWSPowerShell.NetCore	
Cmdlet	New-CWEPartnerEventSource	3.3.563.1
	AWSPowerShell.NetCore	
Cmdlet	Remove-CWEEventBus	3.3.563.1
	AWSPowerShell.NetCore	
Cmdlet	Remove-CWEPartnerEventSource	3.3.563.1
	AWSPowerShell.NetCore	
Cmdlet	Remove-CWEPermission	3.3.563.1
	AWSPowerShell.NetCore	
Cmdlet	Remove-CWEResourceTag	3.3.563.1
	AWSPowerShell.NetCore	
Cmdlet	Remove-CWERule	3.3.563.1
	AWSPowerShell.NetCore	
Cmdlet	Remove-CWETarget	3.3.563.1
	AWSPowerShell.NetCore	
Cmdlet	Test-CWEEventPattern	3.3.563.1
	AWSPowerShell.NetCore	
Cmdlet	Write-CWEEvent	3.3.563.1
	AWSPowerShell.NetCore	
Cmdlet	Write-CWEPartnerEvent	3.3.563.1
	AWSPowerShell.NetCore	
Cmdlet	Write-CWEPermission	3.3.563.1
	AWSPowerShell.NetCore	
Cmdlet	Write-CWERule	3.3.563.1
	AWSPowerShell.NetCore	
Cmdlet	Write-CWETarget	3.3.563.1
	AWSPowerShell.NetCore	

Cmdlet 命名和別名

AWS Tools for PowerShell 中每個服務的 cmdlet 是根據服務 AWS SDK 提供的方法。但是，由於 PowerShell 的強制命名慣例，Cmdlet 的名稱可能會和 API 呼叫名稱，或其根據的方法名稱不同。例如，Get-EC2Instance cmdlet 是根據 Amazon EC2 DescribeInstances 方法。

在某些情況下，cmdlet 名稱可能與方法名稱類似，但實際執行的功能則不同。例如，Amazon S3 `GetObject` 方法會擷取 Amazon S3 物件。然而，`Get-S3Object` cmdlet 會回傳 Amazon S3 物件的資訊，而非物件本身。

```
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
```

若要使用取得 S3 物件 AWS Tools for PowerShell，請執行 `Read-S3Object` cmdlet：

```
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

Cmdlet 的 AWS Cmdlet 說明提供 Cmdlet 所依據的 AWS SDK API 名稱。
如需標準 PowerShell 動詞及其含意的詳細資訊，請參閱[核准的 PowerShell 命令動詞](#)。

當您新增 `-Terminate` 參數時，所有使用動詞的 AWS cmdlet – `Remove` 和 `Stop-EC2Instance` cmdlet – 在繼續之前提示確認。若要略過確認，可在命令裡加入 `-Force` 參數。

Important

AWS cmdlet 不支援 `-WhatIf` 切換。

Aliases

的安裝會 AWS Tools for PowerShell 安裝別名檔案，其中包含許多 AWS cmdlet 的別名。這些別名可能較 cmdlet 名稱更為直觀。例如，服務名稱和 AWS SDK 方法名稱會取代某些別名中的 PowerShell 動詞和名詞。EC2-DescribeInstances 別名就是一個範例。

其他別名即使未遵從標準 PowerShell 慣例，但其使用的動詞可能更詳細描述實際操作。例如，別名檔案會將別名 Get-S3Content 對應至 cmdlet Read-S3Object。

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

別名檔案位於 AWS Tools for PowerShell 安裝目錄中。若要將別名載入到您的環境，請對該檔案使用 dot-source 命令。以下是 Windows 的範例。

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

針對 Linux 或 macOS shell，它看起來會如下：

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

若要顯示所有 AWS Tools for PowerShell 別名，請執行下列命令。這個命令會使用 PowerShell Where-Object Cmdlet 的 `? Source` 屬性來只篩選來自 AWSPowerShell.NetCore 模組的別名。

```
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 AWSPowerShell	Add-ESTag	3.3.343.0
Alias AWSPowerShell	Add-MLTag	3.3.343.0
Alias AWSPowerShell	Clear-AWSCredentials	3.3.343.0
Alias AWSPowerShell	Clear-AWSDefaults	3.3.343.0
Alias AWSPowerShell	Dismount-ASInstances	3.3.343.0
Alias AWSPowerShell	Edit-EC2Hosts	3.3.343.0
Alias AWSPowerShell	Edit-RSClusterIamRoles	3.3.343.0
Alias AWSPowerShell	Enable-ORGAllFeatures	3.3.343.0
Alias AWSPowerShell	Find-CTEvents	3.3.343.0
Alias AWSPowerShell	Get-ASACases	3.3.343.0
Alias AWSPowerShell	Get-ASAccountLimits	3.3.343.0
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	Get-CFCloudFrontOriginAccessIdentities	3.3.343.0
AWSPowerShell		
Alias	Get-CFDistributions	3.3.343.0
AWSPowerShell		
Alias	Get-CFGConfigRules	3.3.343.0
AWSPowerShell		
Alias	Get-CFGConfigurationRecorders	3.3.343.0
AWSPowerShell		
Alias	Get-CFGDeliveryChannels	3.3.343.0
AWSPowerShell		
Alias	Get-CFInvalidations	3.3.343.0
AWSPowerShell		
Alias	Get-CFNAccountLimits	3.3.343.0
AWSPowerShell		
Alias	Get-CFNStackEvents	3.3.343.0
AWSPowerShell		
...		

若要在這個檔案中加入您自己的別名，您可能需要將 PowerShell 的 `$MaximumAliasCount` [偏好變數](#) 提高為大於 5500 的值。預設值為 4096；最多可以提高到 32768。若要進行這項動作，請執行下列命令。

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

若要確認變更成功，可輸入變數名稱以顯示其目前的值。

```
PS > $MaximumAliasCount
32768
```

中的管道、輸出和反覆運算 AWS Tools for PowerShell

管道

PowerShell 鼓勵使用者將 cmdlet 連接到[管道](#)，將一個 cmdlet 的輸出引導至下一個 cmdlet 的輸入。下列範例顯示使用時的此行為 AWS Tools for PowerShell。命令會取得並停止目前預設區域中的所有 Amazon EC2 執行個體。

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

Cmdlet 輸出

为了更好地支援管道，預設適用於 .NET 的 AWS SDK 可能會捨棄來自回應的一些資料。AWS Tools for PowerShell Cmdlet 的輸出不會重塑，以包含服務回應和結果執行個體做為所發出集合物件上的 Note 屬性。而是改為對於發出單一集合做為輸出的這些呼叫，現在將集合列舉到 PowerShell 管道。這表示 SDK 回應和結果資料不能存在於管道中，因為沒有包含可附加的集合物件。

雖然大多數使用者可能不需要此資料，但它對於診斷用途很有用，因為您可以查看 Cmdlet 發出的基礎 AWS 服務呼叫傳送和接收的內容。從 AWS Tools for PowerShell V4 開始，Cmdlet 可以使用 `-Select *` 參數和引數傳回整個服務回應。

Note

在 V4 AWS Tools for PowerShell 之前的版本中，`$AWSHistory` 引進了一個名為的工作階段變數，可維護 AWS cmdlet 調用的記錄以及每次調用收到的服務回應。在 Tools for PowerShell 的 V4 中，此工作階段變數已棄用，有利於 `-Select *` 參數和引數，可用於傳回整個服務回應。本主題說明此參數。

Note

這是預覽版本之服務的發行前版本文件。內容可能變動。

`$AWSHistory` 變數將在 V5 中移除 AWS Tools for PowerShell。如需詳細資訊，請參閱部落格文章 [關於即將推出 AWS 的 Tools for PowerShell 主要版本 5 的通知](#)。

若要說明如何傳回回應中的所有資料，請考慮下列範例。

第一個範例只會傳回 Amazon S3 儲存貯體的清單。這是預設行為。

```
PS > Get-S3Bucket
```

CreationDate	BucketName
-----	-----
9/22/2023 10:54:35 PM	amzn-s3-demo-bucket1
9/22/2023 11:04:37 AM	amzn-s3-demo-bucket2
9/22/2023 12:54:34 PM	amzn-s3-demo-bucket3

第二個範例會傳回適用於 .NET 的 AWS SDK 回應物件。因為 `-Select *` 已指定，所以輸出會包含整個 API 回應，其中包含 Buckets 屬性中的儲存貯體集合。在此範例中，`Format-ListCmdlet` 並非絕對必要，但存在以確保顯示所有屬性。

```
PS > Get-S3Bucket -Select * | Format-List
```

```
LoggedAt           : 10/1/2023 9:45:52 AM
Buckets             : {amzn-s3-demo-bucket1, amzn-s3-demo-bucket2,
                      amzn-s3-demo-bucket3}
Owner               : Amazon.S3.Model.Owner
ContinuationToken   :
ResponseMetadata    : Amazon.Runtime.ResponseMetadata
ContentLength       : 0
HttpStatusCode      : OK
```

透過分頁資料反覆運算

下列各節說明各種可能的反覆運算類型。

自動反覆運算

對於對指定呼叫或支援可分頁結果集施加預設最大傳回物件數量的服務 APIs，大多數 cmdlet 會實作自動反覆運算，這會啟用預設行為 "page-to-completion"。在此案例中，Cmdlet 會代表您進行任意數量的呼叫，將完整的資料集傳回管道。

在下列使用 `Get-S3Object` cmdlet 的範例中，`$result` 變數包含名為 `之儲存貯體中每個金鑰` 的 `S3Object` 執行個體 `amzn-s3-demo-bucket1`，這可能是非常大型的資料集。

```
PS > $result = Get-S3Object -BucketName amzn-s3-demo-bucket1
```

下列範例會將自動反覆運算期間每個頁面的結果數目，從預設值 1000 減少為 500。此範例會執行兩倍的自動重複呼叫，因為每次呼叫只會傳回一半的結果。

```
PS > $result = Get-S3Object -BucketName amzn-s3-demo-bucket1 -MaxKey 500
```

Note

在 AWS Tools for PowerShell V4 中，某些分頁操作的 cmdlet 不會實作自動反覆運算。如果 cmdlet 沒有 `-NoAutoIteration` 參數，將在下一節中討論，則不會實作自動反覆運算。

停用自動反覆運算

如果您希望 Tools for PowerShell 僅傳回資料第一頁，您可以新增 `-NoAutoIteration` 參數，以防止傳回其他頁面的資料。

下列範例使用 `-NoAutoIteration` 和 `-MaxKey` 參數，將傳回的 `S3Object` 執行個體數量限制為不超過儲存貯體中找到的前 500 個執行個體。

```
PS > $result = Get-S3Object -BucketName amzn-s3-demo-bucket1 -MaxKey 500 -  
NoAutoIteration
```

若要判斷是否有更多可用但未傳回的資料，請使用 `-Select *` 參數和引數，並檢查下一個權杖屬性中是否有值。

`$true` 如果儲存貯體中有 500 個以上的物件，`$false` 則以下範例會傳回。

```
PS > $result = Get-S3Object -BucketName amzn-s3-demo-bucket1 -MaxKey 500 -  
NoAutoIteration -Select *  
PS > $null -eq $result.NextMarker
```

Note

下一個權杖回應屬性和 `cmdlet` 參數的名稱會因 `cmdlet` 而異。如需詳細資訊，請參閱每個 `cmdlet` 的說明文件。

手動反覆運算

下列範例使用 `do` 迴圈傳回儲存貯體中的所有 S3 物件，它會在每次反覆運算後評估條件。`do` 迴圈會執行反覆運算，直到 `Get-S3Object $result.NextMarker` 設定為 `$null` 為止，表示不再有分頁資料。迴圈的輸出會指派給 `$s3objects` 變數。

```
$s3objects = do  
{  
    $splatParams = @{  
        BucketName = 'amzn-s3-demo-bucket1'  
        MaxKey = 500  
        Marker = $result.NextMarker  
        NoAutoIteration = $true
```

```
        Select = '*'
    }
    $result = Get-S3Object @splatParams

    $result.S3Objects
}
while ($null -ne $result.NextMarker)
```

此範例使用 PowerShell [潑濺](#)，以避免因為內嵌宣告參數和引數而造成的長程式碼。

憑證和設定檔解析

憑證搜尋順序

當您執行命令時，AWS Tools for PowerShell 會依下列順序搜尋登入資料。當它找到可用的憑證時就會停止。

1. 在命令列中內嵌為參數的文字憑證。

強烈建議您使用描述檔，不要在命令列中放置文字憑證。

2. 指定的描述檔名稱或描述檔位置。

- 如果您只指定設定檔名稱，命令會在 AWS SDK 存放區中尋找指定的設定檔，如果不存在，則會從預設位置的 AWS 共用登入資料檔案中尋找指定的設定檔。
- 如果您只指定描述檔位置，命令會在該憑證檔案中尋找 default 描述檔。
- 如果同時指定名稱和位置，命令就會在該憑證檔案中尋找指定的描述檔。

如果找不到指定的描述檔或位置，命令會擲出例外狀況。只有在您未指定描述檔或位置時，搜尋才會繼續執行下列步驟。

3. -Credential 參數指定的憑證。

4. 工作階段描述檔 (如果存在)。

5. 依以下順序使用預設描述檔：

- a. AWS SDK 存放區中的 default 設定檔。
- b. AWS 共用登入資料檔案中的 default 設定檔。
- c. AWS SDK 存放區中的 AWS PS Default 設定檔。

6. 如果命令是在設定使用 IAM 角色的 Amazon EC2 執行個體上執行，就會從執行個體描述檔存取 EC2 執行個體的暫時憑證。

如需使用適用於 Amazon EC2 執行個體的 IAM 角色詳細資訊，請參閱 [適用於 .NET 的 AWS SDK](#)。

如果此搜尋無法找到指定的憑證，命令會擲出例外狀況。

有關使用者和角色的其他資訊

若要在上執行 Tools for PowerShell 命令 AWS，您需要擁有一些適合您任務的使用者、許可集和服務角色組合。

您建立的特定使用者、許可集合和服務角色，以及使用這些角色的方式，將視您的需求而定。以下提供一些額外的資訊，幫助您了解可能使用它們的原因，以及建立的方法。

使用者和許可集合

雖然可以使用具有長期憑證的 IAM 使用者帳戶來存取 AWS 服務，但這不再是最佳實務，應該避免使用。即使在開發期間，最佳實務是在中建立使用者和許可集，AWS IAM Identity Center 並使用身分來源提供的臨時憑證。

若是開發環境，您可以使用您在 [設定工具身分驗證](#) 中建立或提供給您的使用者。如果您有適當的 AWS Management Console 許可，您也可以為該使用者建立具有最低權限的不同許可集，或為開發專案特別建立新的使用者，為許可集提供最低權限。您選擇的行動方式 (如有的話) 取決於您的情況。

如需有關這些使用者和許可集合，以及如何建立它們的詳細資訊，請參閱 AWS SDK 和工具參考指南中的 [身分驗證和存取](#) 和 AWS IAM Identity Center 使用者指南中的 [入門](#)。

服務角色

您可以設定 AWS 服務角色，以代表使用者存取 AWS 服務。如果有多人將遠端執行您的應用程式，則此類型的存取是適當的；例如，在您為此目的建立的 Amazon EC2 執行個體上。

建立服務角色的程序會根據情況而有所不同，但基本上如下所示。

1. 登入 AWS Management Console，並在 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。
2. 選擇 Roles (角色)，然後選擇 Create role (建立角色)。
3. 選擇 AWS 服務，尋找並選取 EC2 (範例)，然後選擇 EC2 使用案例 (範例)。

4. 選擇下一步，並為應用程式將使用 AWS 的服務選取[適當的政策](#)。

 Warning

請勿選擇 AdministratorAccess 政策，因為該政策會啟用您帳戶中幾乎所有內容的讀取和寫入許可。

5. 選擇下一步。輸入角色名稱、說明，以及您想要的任何標籤。

您可以在 [IAM 使用者指南](#) 中的 [使用 AWS 資源標籤控制存取](#) 中找到標籤的相關資訊。

6. 選擇建立角色。

您可以在 [IAM 使用者指南](#) 中的 [IAM 身分 \(使用者、使用者群組和角色\)](#) 中找到有關 IAM 角色的進階資訊。在 [IAM 角色](#) 主題中尋找有關角色的詳細資訊。

使用舊版憑證

本節中的主題提供有關在不使用 AWS IAM Identity Center 的情況下使用長期或短期憑證資訊。

 Warning

為避免安全風險，在開發專用軟體或使用真實資料時，請勿使用 IAM 使用者進行身分驗證。相反地，搭配使用聯合功能和身分提供者，例如 [AWS IAM Identity Center](#)。

 Note

這些主題中的資訊適用於您需要手動取得及管理短期或長期憑證的情況。有關短期和長期憑證的其他資訊，請參閱 AWS SDK 和工具參考指南中的 [其他驗證方法](#)。

如需最佳安全實務，請使用 AWS IAM Identity Center，如中所述 [設定工具身分驗證](#)。

憑證的重要警告和指引

憑證警告

- 請勿使用您帳戶的根登入資料來存取 AWS 資源。這些登入資料可讓未管制的帳戶存取和很難撤銷這些帳戶。

- 請勿在命令或指令碼中放置常值存取金鑰或憑證資訊。如果這樣做，則會造成意外暴露您的憑證的風險。
- 請注意，存放在共用 AWS credentials 檔案中的任何登入資料都會以純文字儲存。

安全管理憑證的其他指引

如需如何安全管理 AWS 登入資料的一般討論，請參閱《》中的[AWS 安全登入資料](#)[AWS 一般參考](#)和《[IAM 使用者指南](#)》中的[安全最佳實務和使用案例](#)。除了這些討論之外，請考慮下列事項：

- 建立其他使用者 (例如 IAM Identity Center 中的使用者)，並使用其憑證，而不是使用您的 AWS 根使用者憑證。如有必要，其他使用者的憑證可以被撤銷，或本質上是臨時的。此外，您可以將政策套用至每個使用者，以便僅存取特定資源和動作，從而採取最低權限許可的立場。
- 使用適用於 Amazon Elastic Container Service (Amazon ECS) 任務的[任務 IAM 角色](#)。
- 使用在 Amazon EC2 執行個體上執行的應用程式的 [IAM 角色](#)。

主題

- [使用 AWS 登入資料](#)
- [中的共用登入資料 AWS Tools for PowerShell](#)

使用 AWS 登入資料

每個 AWS Tools for PowerShell 命令必須包含一組 AWS 登入資料，用於以密碼編譯方式簽署對應的 Web 服務請求。您可以依命令、工作階段或針對所有工作階段指定憑證。

Warning

為避免安全風險，在開發專用軟體或使用真實資料時，請勿使用 IAM 使用者進行身分驗證。相反地，搭配使用聯合功能和身分提供者，例如 [AWS IAM Identity Center](#)。

Note

本主題中的資訊適用於您需要手動取得及管理短期或長期憑證的情況。有關短期和長期憑證的其他資訊，請參閱 AWS SDK 和工具參考指南中的[其他驗證方法](#)。

如需最佳安全實務，請使用 AWS IAM Identity Center，如 中所述[設定工具身分驗證](#)。

做為避免公開憑證的最佳實務，請勿在命令中輸入文字憑證。請為您要使用的每一組憑證建立一個描述檔，然後將描述檔存放在兩個憑證存放區的其中一個。在命令中依名稱指定正確的描述檔，AWS Tools for PowerShell 就會擷取相關聯的憑證。如需如何安全管理 AWS 憑證的一般討論，請參閱《》中的[管理 AWS 存取金鑰的最佳實務](#) Amazon Web Services 一般參考。

Note

您需要 AWS 帳戶才能取得登入資料並使用 AWS Tools for PowerShell。若要建立 AWS 帳戶，請參閱 AWS 帳戶管理 參考指南中的[入門：您是第一次 AWS 使用嗎？](#)。

主題

- [憑證存放區位置](#)
- [管理描述檔](#)
- [指定憑證](#)
- [憑證搜尋順序](#)
- [在 AWS Tools for PowerShell Core 中處理憑證](#)

憑證存放區位置

AWS Tools for PowerShell 可以使用兩個登入資料存放區中的任一個：

- 軟體 AWS 開發套件存放區會加密您的登入資料，並將其存放在您的主資料夾中。在 Windows 中，此存放區位於：C:\Users*username*\AppData\Local\AWSToolkit\RegisteredAccounts.json。

[適用於 .NET 的 AWS SDK](#) 和 [Toolkit for Visual Studio](#) 也可以使用 AWS 開發套件存放區。

- 共用憑證檔案也位在您的主資料夾中，但以純文字形式存放憑證。

根據預設，憑證檔案存放在這裡：

- 在 Windows 上：C:\Users*username*\.aws\credentials
- 在 Mac/Linux 上：~/.aws/credentials

AWS SDKs 和 AWS Command Line Interface 也可以使用登入資料檔案。如果您在 AWS 使用者內容之外執行指令碼，請確定包含您登入資料的檔案會複製到所有使用者帳戶（本機系統和使用者）都可以存取您登入資料的位置。

管理描述檔

設定檔可讓您使用參考不同的登入資料集 AWS Tools for PowerShell。您可以使用 AWS Tools for PowerShell cmdlet 來管理 AWS SDK 存放區中的設定檔。您也可以使用 [Toolkit for Visual Studio](#)，或以程式設計方式使用 [適用於 .NET 的 AWS SDK](#)，來管理 AWS 開發套件存放區中的描述檔。如需如何管理登入資料檔案中設定檔的指示，請參閱[管理 AWS 存取金鑰的最佳實務](#)。

新增新的描述檔

若要將新設定檔新增至 AWS SDK 存放區，請執行命令 `Set-AWSCredential`。它會將您的存取金鑰和秘密金鑰存放在您指定之描述檔名稱下的預設憑證檔案中。

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

- `-AccessKey`– 存取金鑰 ID。
- `-SecretKey`– 私密金鑰。
- `-StoreAs`– 描述檔名稱，必須是唯一的。若要指定預設描述檔，請使用名稱 `default`。

更新描述檔

AWS SDK 存放區必須手動維護。如果您稍後變更服務上的憑證 (例如使用 [IAM 主控台](#))，使用本機存放的憑證執行命令會失敗，並出現下列錯誤訊息：

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

您可以透過重複描述檔的 `Set-AWSCredential` 命令並將它傳遞到新的存取金鑰和私密金鑰，來更新描述檔。

列出描述檔

您可以使用以下命令檢查目前的名稱清單。在此範例中，名為 Shirley 的使用者可以存取三個描述檔，這些描述檔都儲存在共用憑證檔案 (~/.aws/credentials) 中。

```
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

移除描述檔

若要移除您不再需要的描述檔，請使用以下命令。

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

-ProfileName 參數會指定您要刪除的描述檔。

已移除的命令 [Clear-AWSCredential](#) 仍適用於回溯相容性，但最好使用 Remove-AWSCredentialProfile。

指定憑證

有幾種方式可以指定憑證。偏好的方法是識別設定檔，而不是將常值登入資料納入命令列。會使用[登入資料](#)搜尋順序中所述的搜尋順序來 AWS Tools for PowerShell 尋找設定檔。

在 Windows 上，存放在 AWS SDK 存放區中的 AWS 憑證會使用登入的 Windows 使用者身分加密。它們無法使用其他帳戶解密，也無法在非最初建立帳戶的裝置上使用。若要使用其他使用者的憑證執行任務（例如，有排程任務要執行的使用者帳戶），請如上節所述設定憑證描述檔，以便以該使用者身分登入電腦時可以使用其憑證。以任務執行使用者身分登入，完成憑證設定步驟，並建立適用於該使用者的描述檔。登出後，再使用您自己的憑證登入，以設定排程任務。

Note

使用 -ProfileName 通用參數來指定描述檔。此參數等同於舊版中的 -StoredCredentials 參數 AWS Tools for PowerShell。如需回溯相容性，-StoredCredentials 仍然受支援。

預設描述檔 (建議)

如果登入資料存放在名為 `default` 的設定檔中，AWS SDKs和管理工具可以自動在本機電腦上找到您的登入資料 `default`。例如，如果您在本機電腦上有名為 `default` 的描述檔，就不必執行 `Initialize-AWSDefaultConfiguration` cmdlet 或 `Set-AWSCredential` cmdlet。這些工具會自動使用存放在該描述檔中的存取金鑰和秘密金鑰資料。若要使用您預設區域 (`Get-DefaultAWSRegion` 的結果) 以外的 AWS 區域，您可以執行 `Set-DefaultAWSRegion` 並指定區域。

如果您的描述檔並未命名為 `default`，但您想將其做為目前工作階段使用的預設描述檔，則執行 `Set-AWSCredential` 以將其設定為預設描述檔。

雖然執行 `Initialize-AWSDefaultConfiguration` 可讓您針對每個 PowerShell 工作階段指定預設描述檔，cmdlet 仍從您的自訂名稱描述檔載入憑證，並以命名描述檔覆寫 `default` 描述檔。

我們建議您不執行 `Initialize-AWSDefaultConfiguration`，除非您在未經執行個體描述檔啟動的 Amazon EC2 執行個體上執行 PowerShell 工作階段，且您想要手動設定憑證描述檔。請注意，此藍本中的憑證描述檔不包含憑證。在 EC2 執行個體上執行 `Initialize-AWSDefaultConfiguration` 所產生的憑證描述檔，不會直接存放憑證，而是指向執行個體中繼資料 (提供自動輪換的暫時憑證)。但是，它卻會存放執行個體的區域。另一個可能需要執行 `Initialize-AWSDefaultConfiguration` 的情況是，您希望避免執行個體正在運作的區域，對其他區域執行呼叫。執行該命令會永久覆寫存放在執行個體中繼資料的區域。

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

Note

預設登入資料包含在 AWS SDK 存放區中的 `default` 設定檔名稱下。命令會以該名稱覆寫任何現有的描述檔。

如果您的 EC2 執行個體經執行個體描述檔執行，PowerShell 即會自動從執行個體描述檔取得 AWS 憑證和區域資訊。你不需要執行 `Initialize-AWSDefaultConfiguration`。不需要從經執行個體描述檔啟動的 EC2 執行個體上執行 `Initialize-AWSDefaultConfiguration` cmdlet，因為它預設會使用 PowerShell 已經使用的相同執行個體描述檔資料。

工作階段描述檔

使用 `Set-AWSCredential` 指定特定工作階段的預設描述檔。此描述檔會覆寫工作階段期間內的任何預設描述檔。如果您想要在自己的工作階段中使用自訂名稱的描述檔，而不是目前的 `default` 描述檔，建議您使用此方法。

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

Note

在 1.1 版以前的 Tools for Windows PowerShell 版本中，Set-AWSCredential cmdlet 無法正常運作，而且會覆寫 "MyProfileName" 指定的描述檔。建議使用較新版本 Tools for Windows PowerShell。

命令描述檔

在個別的命令中，您可以新增 -ProfileName 參數以指定僅適用該一個命令的描述檔。此描述檔會覆寫任何預設或工作階段描述檔，如下列範例所示。

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

Note

當您指定預設或工作階段描述檔時，您也可以新增 -Region 參數來覆寫預設或工作階段區域。如需詳細資訊，請參閱[指定 AWS 區域](#)。以下範例指定預設的描述檔和區域。

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

根據預設，AWS 共用的登入資料檔案會假設在使用者的主資料夾中 (C:\Users\username\.awsWindows 或 Linux ~/.aws 上的)。若要指定不同位置的憑證檔案，請加上 -ProfileLocation 參數並指定憑證檔案路徑。以下範例指定特定命令的非預設憑證檔案。

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

Note

如果您要在通常不會登入 AWS 的時間執行 PowerShell 指令碼，例如，在非正常上班時間以排定的任務來執行 PowerShell 指令碼，當您指定要使用的描述檔時請新增 -ProfileLocation

參數，並將值設定為存放您憑證之檔案的路徑。為了確保您的 AWS Tools for PowerShell 指令碼使用正確的帳戶登入資料執行，每當指令碼在不使用 AWS 帳戶的內容或程序中執行時，您應該新增 `-ProfileLocation` 參數。您也可以將您的憑證檔案複製到本機系統或指令碼用於執行任務的其他帳戶可存取的位置。

憑證搜尋順序

當您執行命令時，AWS Tools for PowerShell 會依下列順序搜尋登入資料。當它找到可用的憑證時就會停止。

1. 在命令列中內嵌為參數的文字憑證。

強烈建議您使用描述檔，不要在命令列中放置文字憑證。

2. 指定的描述檔名稱或描述檔位置。

- 如果您只指定設定檔名稱，命令會在 AWS SDK 存放區中尋找指定的設定檔，如果不存在，則會從預設位置的 AWS 共用登入資料檔案中尋找指定的設定檔。
- 如果您只指定描述檔位置，命令會在該憑證檔案中尋找 default 描述檔。
- 如果同時指定名稱和位置，命令就會在該憑證檔案中尋找指定的描述檔。

如果找不到指定的描述檔或位置，命令會擲出例外狀況。只有在您未指定描述檔或位置時，搜尋才會繼續執行下列步驟。

3. `-Credential` 參數指定的憑證。

4. 工作階段描述檔 (如果存在)。

5. 依以下順序使用預設描述檔：

- a. AWS SDK 存放區中的 default 設定檔。
- b. AWS 共用登入資料檔案中的 default 設定檔。
- c. AWS SDK 存放區中的 AWS PS Default 設定檔。

6. 如果命令是在設定使用 IAM 角色的 Amazon EC2 執行個體上執行，就會從執行個體描述檔存取 EC2 執行個體的暫時憑證。

如需使用適用於 Amazon EC2 執行個體的 IAM 角色詳細資訊，請參閱 [適用於 .NET 的 AWS SDK](#)。

如果此搜尋無法找到指定的憑證，命令會擲出例外狀況。

在 AWS Tools for PowerShell Core 中處理憑證

中的 Cmdlet 在執行時 AWS Tools for PowerShell Core 接受 AWS 存取和私密金鑰或登入資料設定檔的名稱，類似於 AWS Tools for Windows PowerShell。在 Windows 上執行時，這兩個模組都能存取適用於 .NET 的 AWS SDK 憑證存放區檔案 (存放在每個使用者的 AppData\Local\AWSToolkit\RegisteredAccounts.json 檔案)。

此檔案以加密格式存放您的金鑰，而且無法用於不同的電腦。這是 AWS Tools for PowerShell 搜尋登入資料描述檔的第一個檔案，也是 AWS Tools for PowerShell 存放登入資料描述檔的檔案。如需適用於 .NET 的 AWS SDK 登入資料存放區檔案的詳細資訊，請參閱[設定 AWS 登入](#)資料。Tools for Windows PowerShell 模組目前不支援將憑證寫入其他檔案或位置。

兩個模組都可以從其他 AWS SDKs 和所使用的 AWS 共用登入資料檔案讀取設定檔 AWS CLI。在 Windows 上，這個檔案的預設位置是 C:\Users\<userid>\.aws\credentials。在非 Windows 平台上，這個檔案存放在 ~/.aws/credentials。-ProfileLocation 參數可用於指向非預設檔案名稱或檔案位置。

開發套件憑證存放區使用 Windows 加密 API，以加密形式存放您的憑證。這些 APIs 不適用於其他平台，因此 AWS Tools for PowerShell Core 模組只會使用 AWS 共用的登入資料檔案，並支援將新的登入資料設定檔寫入共用的登入資料檔案。

以下使用 Set-AWSCredential cmdlet 的範例指令碼，示範在 Windows 上使用 AWSPowerShell 或 AWSPowerShell.NetCore 模組來處理憑證描述檔的選項。

```
# 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
```

以下範例說明 Linux 或 macOS 作業系統上的 AWSPowerShell.NetCore 模組行為。

```
# 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
```

中的共用登入資料 AWS Tools for PowerShell

適用於 Windows PowerShell 的工具支援使用 AWS 共用的登入資料檔案，類似於 AWS CLI 和其他 AWS SDKs。適用於 Windows PowerShell 的工具現在支援讀取和寫入 basic、session 和 assume role 登入資料設定檔到 .NET 登入資料檔案和 AWS 共用登入資料檔案。此功能透過新的 Amazon.Runtime.CredentialManagement 命名空間來啟用。

Warning

為避免安全風險，在開發專用軟體或使用真實資料時，請勿使用 IAM 使用者進行身分驗證。相反地，搭配使用聯合功能和身分提供者，例如 [AWS IAM Identity Center](#)。

Note

本主題中的資訊適用於您需要手動取得及管理短期或長期憑證的情況。有關短期和長期憑證的其他資訊，請參閱 AWS SDK 和工具參考指南中的 [其他驗證方法](#)。

如需最佳安全實務，請使用 AWS IAM Identity Center，如 [中所述設定工具身分驗證](#)。

已新增至憑證相關 cmdlet、[Initiicize-AWSDefaultConfiguration](#)、[New-AWSCredential](#) 和 [Set-AWSCredential](#) 的下列參數支援共用 AWS 憑證檔案的新設定檔類型和存取權。在服務 cmdlet 中，您可以新增通用參數 `-ProfileName` 來參考您的描述檔。

搭配 AWS Tools for PowerShell 使用 IAM 角色

AWS 共用的登入資料檔案可啟用其他類型的存取。例如，您可以使用 IAM 角色來存取 AWS 資源，而不是 IAM 使用者的長期登入資料。若要執行此作業，您的標準描述檔必須擁有可擔任此角色的許可。當您告訴 AWS Tools for PowerShell 使用指定角色的設定檔時，AWS Tools for PowerShell 會查詢 `SourceProfile` 參數識別的設定檔。這些憑證可用來請求 `RoleArn` 參數所指定角色的暫時憑證。當第三方擔任角色時，您可以選擇性地要求使用 Multi-Factor Authentication (MFA) 裝置或 `ExternalId` 代碼。

參數名稱	描述
<code>ExternalId</code>	使用者定義的外部 ID，以用於擔任角色 (如果角色要求)。通常只有在您將帳戶的存取權委派給第三方時才需要這麼做。假設指派的角色時，第三方必須包含 <code>ExternalId</code> 做為參數。如需詳細資訊，請參閱《IAM 使用者指南》中的 如何在將 AWS 資源的存取權授予第三方時使用外部 ID 。
<code>MfaSerial</code>	MFA 序號，以用於擔任角色 (如果角色要求)。如需詳細資訊，請參閱《IAM 使用者指南》中的 在 AWS 中使用多重要素驗證 (MFA) 。
<code>RoleArn</code>	採用角色憑證時擔任的角色 ARN。如需建立和使用角色的詳細資訊，請參閱《IAM 使用者指南》中的 IAM 角色 。
<code>SourceProfile</code>	採用角色憑證所用的來源描述檔名稱。在此描述檔中找到的憑證會用來擔任 <code>RoleArn</code> 參數指定的角色。

設定擔任角色的描述檔

以下範例示範如何設定可直接擔任 IAM 角色的來源描述檔。

第一個命令會建立角色描述檔參考的來源描述檔。第二個命令會建立要擔任哪個角色的角色描述檔。第三個命令會顯示角色描述檔的憑證。

```
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
```

若要搭配 Tools for Windows PowerShell 服務 cmdlet 使用此角色描述檔，請將 `-ProfileName` 通用參數新增至命令以參考角色描述檔。下列範例使用上一個範例中定義的角色描述檔來存取 [Get-S3Bucket](#) cmdlet。AWS Tools for PowerShell 尋找 中的登入資料 `my_source_profile`、使用這些登入資料來 `AssumeRole` 代表使用者呼叫，然後使用這些臨時角色登入資料來呼叫 `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
```

使用憑證描述檔類型

若要設定憑證描述檔類型，請先了解哪些參數提供描述檔類型所需的資訊。

憑證類型	您必須使用的參數
基本	-AccessKey
這些是 IAM 使用者的長期憑證	-SecretKey
工作階段：	-AccessKey

憑證類型	您必須使用的參數
這些是您手動擷取的 IAM 角色短期憑證，例如直接呼叫 Use-STSRole cmdlet。	-SecretKey
	-SessionToken
角色：	-SourceProfile
這些是 AWS Tools for PowerShell 為您擷取的 IAM 角色短期憑證。	-RoleArn
	選用：-ExternalId
	選用：-MfaSerial

ProfilesLocation 通用參數

您可以使用 -ProfileLocation 寫入共用憑證檔案，以及指示 cmdlet 讀取憑證檔案。加入 -ProfileLocation 參數來控制 Tools for Windows PowerShell 是使用共用憑證檔案或是 .NET 憑證檔案。下表說明參數在 Tools for Windows PowerShell 中的運作方式。

描述檔位置值	描述檔解析行為
null (未設定) 或為空	首先，搜尋 .NET 憑證檔案中是否有指定名稱的描述檔。如果找不到設定檔，請在 搜尋 AWS 共用的登入資料檔案(<i>user's home directory</i>) \.aws\credentials。
AWS 共用登入資料檔案格式的檔案路徑	只搜尋指定的檔案中是否有指定名稱的描述檔。

將憑證儲存到憑證檔案

若要將憑證寫入並儲存在其中一個憑證檔案，請執行 Set-AWSCredential cmdlet。下列範例示範其做法。第一個命令會使用 Set-AWSCredential 和 -ProfileLocation，將存取金鑰和秘密金鑰新增至 -ProfileName 參數指定的描述檔。在第二行，執行 [Get-Content](#) cmdlet 以顯示憑證檔案的內容。

```
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
```

顯示您的憑證描述檔

執行 [Get-AWSCredential](#) cmdlet 並新增 `-ListProfileDetail` 參數以傳回憑證檔案類型和位置，以及描述檔名稱清單。

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

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

移除憑證描述檔

若要移除憑證描述檔，請執行新的 [Remove-AWSCredentialProfile](#) cmdlet。 [Clear-AWSCredential](#) 已移除，但仍適用於回溯相容性。

重要說明

只有 [Initialize-AWSDefaultConfiguration](#)、[New-AWSCredential](#) 和 [Set-AWSCredential](#) 支援角色描述檔的參數。您無法直接在 `Get-S3Bucket -SourceProfile source_profile_name -RoleArn arn:aws:iam::999999999999:role/role_name` 等命令中指定角色參數。這不起作用，因為服務 cmdlet 不直接支援 `SourceProfile` 或 `RoleArn` 參數。您反倒必須將這些參數存放在描述檔中，然後使用 `-ProfileName` 參數呼叫命令。

的功能 AWS Tools for Windows PowerShell

本節中的某些主題提供有關 Tools for Windows PowerShell 功能的資訊，您在建立專案和指令碼時可能需要考慮這些功能。本節中的其他主題提供了一些進階方法的資訊，您可以使用這些方法來設定工具、環境和專案。請務必先[安裝](#)並[設定](#)工具。

如需針對特定 AWS 服務開發軟體以及程式碼範例的詳細資訊，請參閱 [使用 AWS 服務](#)。如需其他程式碼範例，請參閱 [適用於 PowerShell V4 的工具程式碼範例](#)。

主題

- [可觀測性](#)

可觀測性

可觀測性是從系統發出的資料推斷系統目前狀態的程度。發出的資料通常稱為遙測。如需使用 AWS 服務時遙測的其他資訊，請參閱《[適用於 .NET 的 AWS SDK 開發人員指南](#)》中的[可觀測性](#)。

下列程式碼顯示如何在 中啟用可觀測性的範例 AWS Tools for PowerShell。

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

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

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

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

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

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

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

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

Set-Location -Path $telemetryProjectPath

dotnet new classlib

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

$classContent = @"
using OpenTelemetry;
using OpenTelemetry.Resources;
using OpenTelemetry.Trace;

namespace Example.Telemetry;

public class AWSToolsForPowerShellTelemetry
{
    public static void InitializeAWSInstrumentation()
    {
        Sdk.CreateTracerProviderBuilder()
            .ConfigureResource(e => e.AddService("$ServiceName"))
    }
}
```



```

        .AddAWSInstrumentation()
        // Exporters vary so options might need to be changed or omitted.
        .AddOtlpExporter(options =>
        {
            options.Endpoint = new Uri("$telemetryEndPoint");
            options.Headers = "$telemetryHeaders";
        })
        .Build();
    }
}
"@

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

dotnet build
dotnet publish -c Release

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

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

# Load assemblies for the telemetry project, excluding the AWS SDK for .NET assemblies
# that were already loaded by importing AWS Tools for PowerShell modules.
$dlls = (Get-ChildItem $dllsPath -Filter *.dll -Recurse ).FullName
$AWSSDKAssembliesAlreadyLoaded =
    [Threading.Thread]::GetDomain().GetAssemblies().Location | Where-Object {$_ -like
        '*AWSSDK*' } | Split-Path -Leaf
$dlls.Where{$AWSSDKAssembliesAlreadyLoaded -notcontains ($_ | Split-Path -
    Leaf)}.ForEach{Add-Type -Path $_}

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

<#
    Now telemetry will be exported for AWS Tools for PowerShell cmdlets
    that are invoked directly or indirectly.
    Execute this cmdlet or execute your own PowerShell script.

```

```
#>
```

```
Get-DDBTable -TableName 'DotNetTests-HashTable' -Region 'us-east-1'
```

在 中使用 AWS 服務 AWS Tools for PowerShell

本節提供使用 AWS Tools for PowerShell 存取 AWS 服務的範例。這些範例有助於示範如何使用 cmdlet 來執行實際 AWS 任務。這些範例會依賴 Tools for PowerShell 所提供的 cmdlet。若要查看有哪些 cmdlet 可用，請參閱 [AWS Tools for PowerShell cmdlet 參考](#)。

PowerShell 檔案串連編碼

編輯現有 AWS Tools for PowerShell 檔案或記錄中的某些 cmdlet AWS。範例為 Edit-R53ResourceRecordSet，這會呼叫適用於 Amazon Route 53 的 [ChangeResourceRecordSets](#) API。

當您在 PowerShell 5.1 或較舊版本中編輯或串連檔案時，PowerShell 會以 UTF-16 而非 UTF-8 輸出編碼。這可能增加不必要的字元，並產生無效的結果。十六進位編輯器可能展現不必要的字元。

為了避免將檔案輸出轉換為 UTF-16，您可以將命令輸送到 PowerShell 的 Out-File cmdlet 並指定 UTF-8 編碼，如以下範例所示。

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

如果您是從 PowerShell 主控台執行 AWS CLI 命令，則適用相同的行為。您可以在 Out-File PowerShell 主控台中將 AWS CLI 命令的輸出輸送至。其他 cmdlet，例如 Export-Csv 或 Export-Clixml，也有一個 Encoding 參數。如需具有 Encoding 參數以及可讓您更正串連檔案輸出編碼的 cmdlet 完整清單，請執行下列命令：

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

Note

PowerShell 6.0 和更新版本 (包括 PowerShell Core) 會自動保留 UTF-8 編碼以供串連檔案輸出。

Powershell 工具傳回的物件

為了在原生 PowerShell 環境中 AWS Tools for PowerShell 更實用，AWS Tools for PowerShell Cmdlet 傳回的物件是 .NET 物件，而不是通常從 AWS SDK 中對應 API 傳回的 JSON 文字物件。

例如，`Get-S3Bucket` 會發出 Buckets 集合，而非 Amazon S3 JSON 回應物件。Buckets 集合可以放置在 PowerShell 管道中，並以適當的方式與之互動。同樣地，`Get-EC2Instance` 會發出 Reservation .NET 物件 集合，而非 `DescribeEC2Instances` JSON 結果物件。此行為是設計而成，可讓 AWS Tools for PowerShell 體驗與慣用的 PowerShell 更一致。

如有需要，您可取得實際的服務回應。它們會儲存為傳回物件上的 `note` 屬性。而透過使用 `NextToken` 欄位支援分頁的 API 動作，也會連接為 `note` 屬性。

Amazon EC2

本節會逐步講解啟動 Amazon EC2 執行個體所需的步驟，其包括以下作業：

- 擷取 Amazon Machine Image (AMI) 的清單。
- 建立 SSH 驗證的金鑰對。
- 建立並設定 Amazon EC2 安全群組。
- 啟動執行個體，並擷取該執行個體的相關資訊。

Amazon Simple Storage Service (Amazon S3)

本節會逐步講解建立靜態網站並託管於 Amazon S3 所需的步驟，其將示範以下作業：

- 建立與刪除 Amazon S3 儲存貯體。
- 將檔案以物件形式上傳至 Amazon S3 儲存貯體。
- 刪除 Amazon S3 儲存貯體中的物件。
- 將 Amazon S3 儲存貯體指定為網站。

AWS Lambda 而且 AWS Tools for PowerShell

本節提供 AWS Lambda Tools for PowerShell 模組的簡短概觀，並說明設定模組所需的步驟。

Amazon SNS 和 Amazon SQS

本節會逐步講解將 Amazon SQS 佇列訂閱至 Amazon SNS 主題所需的步驟。其將示範以下作業：

- 建立 Amazon SNS 主題。

- 建立 Amazon SQS 佇列。
- 將佇列訂閱至 主題。
- 傳送訊息至主題。
- 從佇列接收訊息。

CloudWatch

您可以善用本節提供的範例，藉此了解如何將自訂資料發布至 CloudWatch。

- 發布自訂指標至 CloudWatch 儀表板。

另請參閱

- [開始使用 AWS Tools for Windows PowerShell](#)

主題

- [Amazon S3 和 Tools for Windows PowerShell](#)
- [Amazon EC2 與 Tools for Windows PowerShell](#)
- [AWS Lambda 而且 AWS Tools for PowerShell](#)
- [Amazon SQS、Amazon SNS 和 Tools for Windows PowerShell](#)
- [來自的 CloudWatch AWS Tools for Windows PowerShell](#)
- [在 cmdlets 中使用 ClientConfig 參數](#)

Amazon S3 和 Tools for Windows PowerShell

在本節中，我們使用 Amazon S3 和 CloudFront AWS Tools for Windows PowerShell 建立靜態網站。在過程中，我們展示這些服務的數個常見任務。本演練是根據[託管靜態網站](#)入門指南所建立，該指南描述使用 [AWS 管理主控台](#) 的類似程序。

此處列出的命令假設您已經設定 PowerShell 工作階段的預設憑證與預設區域；所以，cmdlet 並不會呼叫憑證和區域。

Note

目前沒有適用於重新命名儲存貯體或物件的 Amazon S3 API，因此，沒有可執行此任務的單一 Tools for Windows PowerShell cmdlet。若要重新命名 S3 中的物件，建議您執行 [Copy-S3Object](#) cmdlet 使用新名稱複製物件，然後執行 [Remove-S3Object](#) cmdlet 刪除原始物件。

另請參閱

- [在中使用 AWS 服務 AWS Tools for PowerShell](#)
- [託管於 Amazon S3 的靜態網站](#)
- [Amazon S3 主控台](#)

主題

- [建立 Amazon S3 儲存貯體、驗證其區域，或者加以移除](#)
- [設定 Amazon S3 儲存貯體做為網站並啟用記錄](#)
- [將物件上傳至 Amazon S3 儲存貯體](#)
- [刪除 Amazon S3 物件與儲存貯體](#)
- [將內嵌文字內容上傳到 Amazon S3](#)

建立 Amazon S3 儲存貯體、驗證其區域，或者加以移除

使用 `New-S3Bucket` cmdlet 來建立新的 Amazon S3 儲存貯體。以下範例會建立名為 `website-example` 的儲存貯體。儲存貯體的名稱必須在所有區域中為唯一。範例會在 `us-west-1` 區域中建立儲存貯體。

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

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

您可以使用 `Get-S3BucketLocation` cmdlet 驗證儲存貯體所在的區域。

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

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

當你完成本教學後，您就可以使用下面的程式碼刪除這個儲存貯體。但建議您保留這個儲存貯體，因為我們在後續範例會用到它。

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

請注意，儲存貯體移除程序需要較長時間來完成。如果您嘗試立即重新建立同名的儲存貯體，New-S3Bucket cmdlet 會失敗，直到舊的儲存貯體完全消失為止。

另請參閱

- [在中使用 AWS 服務 AWS Tools for PowerShell](#)
- [Put Bucket \(Amazon S3 服務參考\)](#)
- [AWS Amazon S3 的 PowerShell 區域](#)

設定 Amazon S3 儲存貯體做為網站並啟用記錄

使用 Write-S3BucketWebsite cmdlet 來設定 Amazon S3 儲存貯體做為靜態網站。以下範例指定預設內容網頁的 index.html 名稱以及預設錯誤網頁的 error.html 名稱。請注意，這個 cmdlet 不會建立這些頁面。需將它們[上傳為 Amazon S3 物件](#)。

```
PS > Write-S3BucketWebsite -BucketName website-example -
WebsiteConfiguration_IndexDocumentSuffix index.html -WebsiteConfiguration_ErrorDocument
error.html
RequestId      : A1813E27995FFDDDD
AmazonId2      : T7h1D0eLqA5Q2XfTe8j2q3SL0P3/5XwhUU3RyJBGHU/LnC+CIWLeGgP0MY24xA1I
ResponseStream :
Headers        : {x-amz-id-2, x-amz-request-id, Content-Length, Date...}
Metadata       : {}
ResponseXml    :
```

另請參閱

- [在中使用 AWS 服務 AWS Tools for PowerShell](#)

- [Put Bucket 網站 \(Amazon S3 API 參考\)](#)
- [Put Bucket ACL \(Amazon S3 API 參考\)](#)

將物件上傳至 Amazon S3 儲存貯體

使用 Write-S3Object cmdlet 可從本機檔案系統，將檔案以物件形式上傳至 Amazon S3 儲存貯體。以下範例會建立和上傳兩個簡單的 HTML 檔案至 Amazon S3 儲存貯體，並驗證上傳物件是否存在。-File 的 Write-S3Object 參數指定本機檔案系統中的檔案名稱。-Key 參數指定 Amazon S3 中對應物件的名稱。

Amazon 會從副檔名推斷物件的內容類型，在此例中是「.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
```

```
-----
```

```
index.html
```

```
ErrorDocument
```

```
-----
```

```
error.html
```

固定的 ACL 選項

使用 Tools for Windows PowerShell 指定固定的 ACL 之值，與適用於 .NET 的 AWS SDK 所用的值相同。不過，請注意，這些值都不同於 Amazon S3 Put Object 動作所用的值。Tools for Windows PowerShell 支援下列固定的 ACL：

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

如需有關這些固定 ACL 設定的詳細資訊，請參閱[存取控制清單概觀](#)。

分段上傳的相關注意事項

如果您使用 Amazon S3 API 來上傳大小大於 5 GB 的檔案，您需要使用分段上傳。不過，Tools for Windows PowerShell 提供的 Write-S3Object cmdlet 可以透明的方式處理大於 5 GB 的檔案上傳。

測試網站

此時，您可以使用瀏覽器瀏覽至網站來測試網站。Amazon S3 中託管的靜態網站 URL 遵循標準格式。

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

例如：

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

另請參閱

- [在中使用 AWS 服務 AWS Tools for PowerShell](#)
- [Put 物件 \(Amazon S3 API 參考\)](#)
- [固定的 ACL \(Amazon S3 API 參考\)](#)

刪除 Amazon S3 物件與儲存貯體

本節會說明如何將您在先前章節中所建立的網站予以刪除。您僅需刪除 HTML 檔案的物件，接著刪除該網站的 Amazon S3 儲存貯體即可。

首先，執行 `Remove-S3Object` cmdlet，從 Amazon S3 儲存貯體中刪除 HTML 檔案的物件。

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

經由 Amazon S3 處理請求的方式，系統預期顯示的成品即為 `False` 回應。在此情況下，該回應並不表示發生問題。

您現在可以執行 `Remove-S3Bucket` cmdlet，刪除該網站目前空白的 Amazon S3 儲存貯體。

```
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    :
```

在 1.1 版和更新版本中 AWS Tools for PowerShell，您可以將 `-DeleteBucketContent` 參數新增至 `Remove-S3Bucket`，先刪除指定儲存貯體中的所有物件和物件版本，再嘗試移除儲存貯體本身。根據

儲存貯體中的物件或物件版本數量，系統可能需要花費大量的時間才能完成此操作。若使用 Tools for Windows PowerShell 1.1 版以前的版本，則儲存貯體必須為空，`Remove-S3Bucket` 才能夠刪除它。

Note

除非您新增 `-Force` 參數，否則 會在 cmdlet 執行之前 AWS Tools for PowerShell 提示您進行確認。

另請參閱

- [在中使用 AWS 服務 AWS Tools for PowerShell](#)
- [刪除物件 \(Amazon S3 API 參考\)](#)
- [DeleteBucket \(Amazon S3 API 參考\)](#)

將內嵌文字內容上傳到 Amazon S3

`Write-S3Object` cmdlet 支援將內嵌文字內容上傳至 Amazon S3 的能力。使用 `-Content` 參數 (別名 `-Text`)，您可以指定應上傳到 Amazon S3 的文字型內容，而不需先將它轉換為檔案。參數接受簡單的一行字串，以及包含多行的 `here` 字串。

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

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

PS > # Specifying content from a variable: (note final newline needed to end in-line
here-string)
PS > $x = @"
>> line 1
>> line 2
>> line 3
>> "e
```

```
>>
PS > write-s3object amzn-s3-demo-bucket -key myobject.txt -content $x
```

Amazon EC2 與 Tools for Windows PowerShell

您可以使用 執行與 Amazon EC2 相關的常見任務 AWS Tools for PowerShell。

此處列出的範例命令假設您已經設定 PowerShell 工作階段的預設憑證與預設區域；因此，當我們呼叫 cmdlet 時不包含憑證或區域。如需詳細資訊，請參閱[開始使用 AWS Tools for Windows PowerShell](#)。

主題

- [建立金鑰對](#)
- [使用 Windows PowerShell 建立安全群組](#)
- [使用 Windows PowerShell 來尋找 Amazon Machine Image](#)
- [使用 Windows PowerShell 啟動 Amazon EC2 執行個體](#)

建立金鑰對

以下 New-EC2KeyPair 範例會建立金鑰對，並存放在 PowerShell 變數 \$myPSKeyPair 中

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

將金鑰對物件輸送到 Get-Member Cmdlet，來查看物件的結構。

```
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;}

將金鑰對物件輸送到 `Format-List cmdlet` 以檢視 `KeyName`、`KeyFingerprint` 和 `KeyMaterial` 成員的值。(輸出已截斷以利閱讀)。

```
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...
                   Mz6bt0xPcE7EMeH1wySUP8nouAS9xb1917+Vkd74bN9KmNcPa/Mu...
                   Zyn4vVe0Q5il/MpkrRogHq0B0rigeTeV5Yc3lv00RFFPu0Kz4kcm...
                   w3Jg8dKsWn0p10pX7V3sRC02KgJIbejQUvBFGi50QK9bm4tXBIeC...
                   daxKIAQMtDUdmBDrhR1/YMv8itFe5DiLLbq7Ga+FDcS85NstBa3h...
                   iuskGkcvgWkcFQkLmRHROdpPb+OdFsZtjHZDpMVFMa9tT8EdbkEF...
                   3SrNeqZPsxJJIX0odb3CxLJpg75JU5kyWnb0+sDNVHoJiZCULCr0...
                   GG1LfEgB95KjGIk7zEv2Q7K6s+DHclrDeMZwa7KFNRZuCuX7jssC...
                   x098abxMr3o3TNU6p1ZYRJEQ0oJr0W+kc+/8SWb8NIwflTwhmJEy...
                   1BX9X8WFX/A8VLHrT1e1rKmlkNECgYEAwltkV1p0JAFhz9p7ZFEv...
                   vvVsPaF0Ev9bk9pqhx269PB50x2KokwCagDMMaYvasWobuLmNu/1...
                   1mWrx7KTeQ7W1J30LgxHA1QNMkip9c4Tb3q9vVc3t/fPf8vwfJ8C...
                   63g6N6rk2FkHZX1E62BgbewUd3eZ0S05Ip4VUdvtGcuc8/qa+e5C...
                   KXgyt9n164pMv+VaxfXkZhdLAdY0Khc9TGB9++VM5G5TrD15YJId...
                   gYALEI7m1jJKpHWAES0hiemw5VmKyIZpzGstSJsFstERlAjiETDH...
                   YAtnI4J8dRyp9I7B0V0n3wNfIjk85gi1/00c+j8S65giLAfndWGR...
                   9R9wIkm5BMUcSRRcDy0yuwKBgEbk0nGGSD0ah4HkvrUkepIbUDTD...
                   AnEBM1cXI5UT7BfKInpUihZi59QhgdK/hk0SmWhlZGWikJ5VizBf...
                   drkBr/vTKVRMTi3lVFB7KkIV1xJxC5E/BZ+YdZEpWoCZAoGAC/Cd...
                   TTld5N6opg0XAcQJwzqoGa9ZMwc5Q9f4bfRc67emkw0ZAAwSsvWR...
                   x302duuy7/smTwWwskEWRK5IrUxoMv/VVYaqdzc0ajwieNrb1r7c...
                   -----END RSA PRIVATE KEY-----
```

`KeyMaterial` 成員用於存放金鑰對的私有金鑰。公有金鑰存放在 中 AWS。您無法從 擷取公有金鑰 AWS，但您可以透過比較私有金鑰 `KeyFingerprint` 的 與針對公有金鑰從 傳回 AWS 的 來驗證公有金鑰。

檢視金鑰對的指紋

您可以使用 `Get-EC2KeyPair cmdlet` 來檢視金鑰對的指紋。

```
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
```

存放您的私有金鑰

若要將私有金鑰存放至檔案，請將 `KeyFingerMaterial` 成員輸送至 `Out-File` cmdlet。

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

將私有金鑰寫入檔案時，您必須指定 `-Encoding ascii`。否則，`openssl` 等工具將可能無法正確地讀取檔案。您可以使用如下所示的命令，驗證所產生檔案的格式是否正確：

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

(AWS Tools for PowerShell 或 不包含 `openssl` 工具 適用於 .NET 的 AWS SDK。)

移除您的金鑰對

您需要金鑰對，才能啟動和連線到執行個體。當您使用完金鑰對後，您可以將其移除。若要從中移除公有金鑰 AWS，請使用 `Remove-EC2KeyPair` cmdlet。出現提示時，按下 `Enter` 移除金鑰對。

```
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"):

變數 `$myPSKeyPair` 仍然存在於目前的 PowerShell 工作階段，也仍包含金鑰對資訊。`myPSKeyPair.pem` 檔案也存在。不過，私有金鑰不再有效，因為金鑰對的公有金鑰不再存放於 AWS 中。

使用 Windows PowerShell 建立安全群組

您可以使用 AWS Tools for PowerShell 來建立和設定安全群組。回應為安全群組的 ID。

如果您需要連接到您的執行個體，則必須設定安全群組以允許 SSH 流量 (Linux) 或 RDP 流量 (Windows)。

主題

- [先決條件](#)
- [建立適用於 EC2-VPC 的安全群組](#)

先決條件

您需要電腦的公有 IP 地址 (使用 CIDR 表示法)。您可以透過一項服務來取得本機電腦的公有 IP 地址。例如，Amazon 提供以下服務：<http://checkip.amazonaws.com/> 或 <https://checkip.amazonaws.com/>。若要尋找其他能夠提供您的 IP 地址之服務，請搜尋字詞「what is my IP address」(我的 IP 地址為何)。如果您透過 ISP 或是從防火牆後方進行連線，不具備靜態 IP 地址，則需要找到您的用戶端電腦可以使用的 IP 地址範圍。

Warning

若您指定 `0.0.0.0/0`，您便會啟用來自世界任何 IP 地址的流量。針對 SSH 和 RDP 通訊協定，您可能會將在測試環境中短暫進行此作業視為沒有問題，但用在生產環境則不安全。在生產環境中，請務必僅授權來自適當個別 IP 地址或地址範圍的存取。

建立適用於 EC2-VPC 的安全群組

Warning

EC2-Classic 在 2022 年 8 月 15 日淘汰。建議您從 EC2-Classic 遷移至 VPC。如需詳細資訊，請參閱部落格文章 [EC2-Classic Networking 正在淘汰 – 以下是如何準備](#)。

以下 `New-EC2SecurityGroup` 範例會新增 `-VpcId` 參數來建立指定 VPC 的安全群組。

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

若要檢視安全群組的初始設定，請使用 `Get-EC2SecurityGroup` cmdlet。在預設情況下，適用於 VPC 的安全群組包含允許所有對外流量的規則。請注意，EC2-VPC 的安全群組不能按名稱引用。

```
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                 : {}

```

若要定義 TCP 連接埠 22 (SSH) 及 TCP 連接埠 3389 上傳入流量的許可，請使用 `New-Object Cmdlet`。以下範例指令碼會定義單一 IP 地址 203.0.113.25/32 在 TCP 連接埠 22 和 3389 的許可。

```

$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 )

```

若要驗證安全群組是否已更新，請再次使用 `Get-EC2SecurityGroup cmdlet`。

```

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              : {}

```

若要檢視傳入規則，您可以從先前命令傳回的集合物件中擷取 `IpPermissions` 屬性。

```

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}
```

使用 Windows PowerShell 來尋找 Amazon Machine Image

啟動 Amazon EC2 執行個體時，您會指定 Amazon Machine Image (AMI) 做為執行個體的範本。不過，AWS Windows AMIs IDs 會經常變更，因為 AWS 會提供具有最新更新和安全性增強功能的新 AMIs。您可使用 [Get-EC2Image](#) 和 [Get-EC2ImageByName](#) Cmdlet 來尋找目前的 Windows AMI 並取得其 ID。

主題

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

Get-EC2Image

Get-EC2Image cmdlet 會擷取您可使用的 AMI 清單。

搭配陣列值 -Owner 使用 amazon, self 參數，讓 Get-EC2Image 僅擷取屬於 Amazon 或您的 AMI。在此內容中，「您」指的是您使用其憑證來呼叫 Cmdlet 的使用者。

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

您可使用 -Filter 參數來限定結果範圍。欲指定篩選條件，請建立類型 Amazon.EC2.Model.Filter 的物件。例如，下列篩選條件只會顯示 Windows AMI。

```
$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
```

下列範例為 cmdlet 回傳的 AMI 之一；上述命令的輸出實際提供許多 AMI 的資訊。

```
Architecture      : x86_64
BlockDeviceMappings : {/dev/sda1, xvdca, xvdcdb, 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

Get-EC2ImageByName Cmdlet 可讓您依據您感興趣的伺服器組態類型，篩選 AWS Windows AMI 的清單。

不帶參數執行時，Cmdlet 會發出目前篩選條件的整組名稱，如下所示：

```
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

```

欲限縮回傳的映像組，請使用 `Names` 參數來指定一個或多個篩選條件名稱。

```
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
```

使用 Windows PowerShell 啟動 Amazon EC2 執行個體

若要啟動 Amazon EC2 執行個體，您需要您在先前章節中建立的金鑰對和安全群組。您也需要 Amazon Machine Image (AMI) 的 ID。如需詳細資訊，請參閱下列文件：

- [建立金鑰對](#)
- [使用 Windows PowerShell 建立安全群組](#)
- [使用 Windows PowerShell 來尋找 Amazon Machine Image](#)

Important

如果您啟動的執行個體不包含在免費方案內，則需要在啟動該執行個體後開始收費，收費根據執行個體的執行時間而定，即使其保持閒置狀態仍會計費。

主題

- [在 VPC 中啟動執行個體](#)
- [在 VPC 中啟動 Spot 執行個體](#)

在 VPC 中啟動執行個體

Warning

EC2-Classic 在 2022 年 8 月 15 日淘汰。建議您從 EC2-Classic 遷移至 VPC。如需詳細資訊，請參閱部落格文章 [EC2-Classic Networking 正在淘汰 – 以下是如何準備](#)。

下列命令會在指定的私有子網路中建立單一 `m1.small` 執行個體。安全群組必須針對指定的子網路有效。

```
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        : {}
```

您的執行個體一開始會處於 `pending` 狀態，但在幾分鐘之後就會進入 `running` 狀態。若要檢視執行個體的相關資訊，請使用 `Get-EC2Instance` cmdlet。如果您有多個執行個體，則可以使用 `Filter` 參數來篩選預留 ID 的結果。首先，建立 `Amazon.EC2.Model.Filter` 類型的物件。接下來，請呼叫使用篩選條件的 `Get-EC2Instance`，然後顯示 `Instances` 屬性。

```
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

```

在 VPC 中啟動 Spot 執行個體

以下範例指令碼會請求指定子網路中的 Spot 執行個體。安全群組必須是您為 VPC 所建立，且包含指定的子網路。

```

$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 而且 AWS Tools for PowerShell

透過使用 [AWSLambdaPSCore](#) 模組，您可以使用 .NET Core 2.1 執行時間在 PowerShell Core 6.0 中開發 AWS Lambda 函數。PowerShell 開發人員可以使用 Lambda 在 PowerShell 環境中管理 AWS 資源和撰寫自動化指令碼。Lambda 中的 PowerShell 支援可讓您執行 PowerShell 指令碼或函數，來回應任何 Lambda 事件，例如 Amazon S3 事件或 Amazon CloudWatch 排定事件。AWSLambdaPSCore 模組是 PowerShell 的個別 AWS 模組；它不是的一部分 AWS Tools for PowerShell，也不會安裝 AWSLambdaPSCore 模組來安裝 AWS Tools for PowerShell。

安裝 AWSLambdaPSCore 模組後，您可以使用任何可用的 PowerShell 或您自行開發的 cmdlet 來編寫無伺服器函數。AWS Lambda Tools for PowerShell 模組包含 PowerShell 型無伺服器應用程式的專案範本，以及發佈專案的工具 AWS。

所有支援 Lambda 的區域皆支援 AWSLambdaPSCore 模組。如需支援區域的詳細資訊，請參閱 [AWS 區域表](#)。

先決條件

您必須先執行以下必要步驟，才能安裝和使用 AWSLambdaPSCore 模組。如需這些步驟的詳細資訊，請參閱《AWS Lambda 開發人員指南》中的[設定 PowerShell 開發環境](#)。

- 安裝正確版本的 PowerShell – Lambda 對 PowerShell 的支援取決於跨平台 PowerShell Core 6.0 版本。您可以在 Windows、Linux、或 Mac 上開發 PowerShell Lambda 函數。如果您尚未安裝此版本 PowerShell，可在 [Microsoft PowerShell 文件網站](#) 中找到指示。
- 安裝 .NET Core 2.1 開發套件 - 由於 PowerShell Core 是以 .NET Core 為基礎建置而成，在 .NET Core 和 PowerShell Lambda 函數兩方面，PowerShell 的 Lambda 支援皆使用相同的 .NET Core 2.1 Lambda 執行時間。Lambda PowerShell 發佈 cmdlet 使用 .NET Core 2.1 開發套件來建立 Lambda 部署套件。可從 [Microsoft 下載中心](#) 取得 .NET Core 2.1 SDK。請務必安裝軟體開發套件，而非執行時間。

安裝 AWSLambdaPSCore 模組

完成先決條件之後，您就可以安裝 AWSLambdaPSCore 模組。在 PowerShell Core 工作階段中執行下列命令。

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

您已可以開始在 PowerShell 中開始開發 Lambda 函數。如需入門詳細資訊，請參閱《AWS Lambda 開發人員指南》中的[以 PowerShell 撰寫 Lambda 函數的程式設計模型](#)。

另請參閱

- [在 AWS 開發人員部落格上宣布對 PowerShell Core 的 Lambda 支援](#)
- [PowerShell Gallery 網站上的 AWSLambdaPSCore 模組](#)
- [設定 PowerShell 開發環境](#)
- [AWS GitHub 上的適用於 Powershell 的 Lambda 工具](#)
- [AWS Lambda 主控台](#)

Amazon SQS、Amazon SNS 和 Tools for Windows PowerShell

本節提供的範例，說明執行以下作業的方法：

- 建立 Amazon SQS 佇列和取得佇列 ARN (Amazon 資源名稱)。
- 建立 Amazon SNS 主題。
- 提供許可給 SNS 主題，以便其可以傳送訊息至佇列。
- 訂閱佇列至 SNS 主題
- 授予 IAM 使用者或 AWS 帳戶發佈至 SNS 主題並從 SQS 佇列讀取訊息的許可。
- 將訊息發佈至主題並讀取來自佇列的訊息以驗證結果。

建立 Amazon SQS 佇列和取得佇列 ARN

下列命令會在您的預設區域中建立 SQS 佇列。輸出會顯示新佇列的 URL。

```
PS > New-SQSQueue -QueueName myQueue
```



```
https://sqs.us-west-2.amazonaws.com/123456789012/myQueue
```

下面的命令會擷取佇列的 ARN。

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

建立 Amazon SNS 主題

下列命令會在您的預設區域中建立 SNS 主題，並傳回新主題的 ARN。

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

提供許可給 SNS 主題

下列範例指令碼會同時建立 SQS 佇列和 SNS 主題，並授與 SNS 主題的許可，以便將訊息傳送到 SQS 佇列：

```
# 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 }
```

訂閱佇列至 SNS 主題

下列命令會訂閱 SNS 主題 myTopic 的佇列 myQueue，並傳回訂閱 ID：

```
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
```

提供許可

以下命令授予對主題 myTopic 執行 sns:Publish 動作的許可

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

以下命令授予對佇列 myQueue 執行 sqs:ReceiveMessage 和 sqs:DeleteMessage 動作的許可

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

驗證結果

下列命令會將訊息發佈至 SNS 主題 myTopic，測試您的新佇列和主題，並傳回 MessageId。

```
PS > Publish-SNSMessage `
```

```
-TopicArn arn:aws:sns:us-west-2:123456789012:myTopic`
```

```
-Message "Have A Nice Day!"
```

```
728180b6-f62b-49d5-b4d3-3824bb2e77f4
```

以下命令會從 SQS 佇列 myQueue 擷取並顯示訊息。

```
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" :
```

```
    "1lE17A2+X0uJZnw3TlgcXz4C4KPLXZxbxoEMIirelh13u/oxkWmz5+9tJKFMns1Z0qQvKxk
```

```
+ExfEZcD5yWt6biVuBb8pyRmZ1b03hUEN13ayv2WQiQT1vpLpM7VEQN5m+hLIiPFcs
```

```
    vYuGkJReV710JWPHnCN
```

```
+qTE2lId2RPkF0eGtLGawTsSPTWEvJdDbLlf7E0zZ0q1niXTUtpsZ8Swx01X3Q06u9i9qBFt0ekJFZNJp6Avu05hIk1b4yo
y0a8Yl9lWp7a7EoWaBn0zhCESe7o
```

```
    kZC6ncBJWphX7KCGVYD0qhVf/5VDgBuv9w8T+higJyvr3WbaSvg==",
```

```
    "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/
```

```
SimpleNotificationService-6aad65c2f9911b05cd53efda11f913f9.pem",
```

```
    "UnsubscribeURL" :
```

```
    "https://sns.us-west-2.amazonaws.com/?
```

```
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-west-2:123456789012:myTopic:22b77de7-
a216-4000-9a23-bf465744ca84"
```

```
  }
```

```
MD5ofBody : 5b5ee4f073e9c618eda3718b594fa257
```

```
MD5ofMessageAttributes :
```

```
MessageAttributes : {}
```

```
MessageId : 728180b6-f62b-49d5-b4d3-3824bb2e77f4
```

```
ReceiptHandle :
```

```
  AQEB2vvk1e5c0KFjeIWJticabkc664yuDEjhucnIOqdVUmie7bX7GiJb17F0enABUgaI2XjEcNPxixhVc/
wfsAJZLNHn18S1bQa0R/kD+Sqa40Ivfj8x3M40h1yM1cVKpYmhAzsYrAwAD5g5FvxNBD6zs
```

```
    +HmXdkax2Wd+9AxrH1QZV5ur1MoByKWwbDbsqoYJTJquCc10gWIak/sBx/
```

```
daBRMTiVQ4GHsrQWMVHtNC14q7Jy/0L2dkmb4dzJfJq0VbFSX1G+u/lrSLpgae+Dfux646y8yFiPFzY4ua4mCF/
SVUn63Spy
```

```
    sHN12776axknhg3j9K/Xwj54DixdsegrnrKoLx+ctI
```

```
+0jzAetBR66Q1VhIoJAq7s0a2Msey0eM/Jjucg6Sr9VUnTWVhV8ErXmotoiEg==
```

來自的 CloudWatch AWS Tools for Windows PowerShell

您可以善用本節提供的 Tools for Windows PowerShell 使用範例，藉此透過該工具將自訂指標資料發佈至 CloudWatch。

此範例會假設您已經設定 PowerShell 工作階段的預設憑證與預設區域；

發布自訂指標至 CloudWatch 儀表板

以下 PowerShell 程式碼會初始化 CloudWatch MetricDatum 物件並將它發佈至服務。瀏覽至 [CloudWatch 主控台](#)，即可查看此操作的結果。

```
$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
```

注意下列事項：

- 日期與時間資訊必須採用國際標準時間 (UTC)，才能夠用來將 \$dat.Timestamp 初始化。
- 加上引號的字串值，或是不加引號的數值皆可用來將 \$dat.Value 初始化；此範例會顯示字串值。

另請參閱

- [在中使用 AWS 服務 AWS Tools for PowerShell](#)
- [AmazonCloudWatchClient.PutMetricData](#) (.NET 開發套件參考)
- [MetricDatum](#) (服務 API 參考)
- [Amazon CloudWatch 主控台](#)

在 cmdlets 中使用 ClientConfig 參數

當您連線至服務時，此 ClientConfig 參數可用來指定特定組態設定。此參數的大部分可能屬性都是在 [Amazon.Runtime.ClientConfig](#) 類別中定義，其會繼承至 APIs。AWS 請參閱 [Amazon.Keyspaces.AmazonKeyspacesConfig](#) 類別以查看簡易的繼承範例。此外，特定服務會

定義其他僅適用於該服務的屬性。請參閱 [Amazon.S3.AmazonS3Config](#) 類別以查看其他屬性的範例，尤其是 ForcePathStyle 屬性。

使用 ClientConfig 參數。

若要使用 ClientConfig 參數，您可以在命令列上將其指定為 ClientConfig 物件，或使用 PowerShell splatting 將參數值集合當做一個單位傳遞給命令。這些方法如下範例所示。這些範例預設 AWS.Tools.S3 模組已安裝並匯入，而且您的 [default] 憑證設定檔具有適當權限。

定義 ClientConfig 物件

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

使用 PowerShell splatting 新增 ClientConfig 屬性

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

Get-S3Object @params
```

使用未定義的屬性

使用 PowerShell 潑濺時，如果您指定不存在的 ClientConfig 屬性，在執行時間之前，AWS Tools for PowerShell 不會偵測到錯誤，此時會傳回例外狀況。依上述修改範例：

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

```
Get-S3Object @params
```

此範例會產生類似下列的例外狀況：

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

指定 AWS 區域

您可以使用 `ClientConfig` 參數來設定 命令 AWS 區域 的。Region (區域) 是透過 `RegionEndpoint` 屬性進行設定。會根據下列優先順序 AWS Tools for PowerShell 計算要使用的區域：

1. `-Region` 參數
2. 在 `ClientConfig` 參數中傳遞的區域
3. PowerShell 工作階段狀態
4. 共用 AWS config 檔案
5. 環境變數
6. Amazon EC2 執行個體中繼資料 (若已啟用)。

適用於 PowerShell V4 的工具程式碼範例

本主題中的程式碼範例說明如何搭配使用 AWS Tools for PowerShell V4 AWS。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他 AWS 服務組合來完成特定任務的程式碼範例。

有些服務包含其他範例類別，示範如何利用服務特定的程式庫或函數。

服務

- [使用 Tools for PowerShell V4 的 ACM 範例](#)
- [使用 Tools for PowerShell V4 的 Application Auto Scaling 範例](#)
- [使用 Tools for PowerShell V4 的 AppStream 2.0 範例](#)
- [使用 Tools for PowerShell V4 的 Aurora 範例](#)
- [使用 Tools for PowerShell V4 的 Auto Scaling 範例](#)
- [AWS Budgets 使用 Tools for PowerShell V4 的範例](#)
- [AWS Cloud9 使用 Tools for PowerShell V4 的範例](#)
- [AWS CloudFormation 使用 Tools for PowerShell V4 的範例](#)
- [使用 Tools for PowerShell V4 的 CloudFront 範例](#)
- [使用 Tools for PowerShell V4 的 CloudTrail 範例](#)
- [使用 Tools for PowerShell V4 的 CloudWatch 範例](#)
- [使用 Tools for PowerShell V4 的 CodeCommit 範例](#)
- [使用 Tools for PowerShell V4 的 CodeDeploy 範例](#)
- [使用 Tools for PowerShell V4 的 CodePipeline 範例](#)
- [使用 Tools for PowerShell V4 的 Amazon Cognito 身分範例](#)
- [AWS Config 使用 Tools for PowerShell V4 的範例](#)
- [使用 Tools for PowerShell V4 的 Device Farm 範例](#)
- [AWS Directory Service 使用 Tools for PowerShell V4 的範例](#)
- [AWS DMS 使用 Tools for PowerShell V4 的範例](#)
- [使用 Tools for PowerShell V4 的 DynamoDB 範例](#)

- [使用 Tools for PowerShell V4 的 Amazon EC2 範例](#)
- [使用 Tools for PowerShell V4 的 Amazon ECR 範例](#)
- [使用 Tools for PowerShell V4 的 Amazon ECS 範例](#)
- [使用 Tools for PowerShell V4 的 Amazon EFS 範例](#)
- [使用 Tools for PowerShell V4 的 Amazon EKS 範例](#)
- [Elastic Load Balancing - 使用 Tools for PowerShell V4 的第 1 版範例](#)
- [Elastic Load Balancing - 使用 Tools for PowerShell V4 的第 2 版範例](#)
- [使用 Tools for PowerShell V4 的 Amazon FSx 範例](#)
- [AWS Glue 使用 Tools for PowerShell V4 的範例](#)
- [AWS Health 使用 Tools for PowerShell V4 的範例](#)
- [使用 Tools for PowerShell V4 的 IAM 範例](#)
- [使用 Tools for PowerShell V4 的 Kinesis 範例](#)
- [使用 Tools for PowerShell V4 的 Lambda 範例](#)
- [使用 Tools for PowerShell V4 的 Amazon ML 範例](#)
- [使用 Tools for PowerShell V4 的 Macie 範例](#)
- [AWS OpsWorks 使用 Tools for PowerShell V4 的範例](#)
- [AWS 價格表 使用 Tools for PowerShell V4 的範例](#)
- [使用 Tools for PowerShell V4 的資源群組範例](#)
- [使用 Tools for PowerShell V4 標記 API 範例的資源群組](#)
- [使用 Tools for PowerShell V4 的 Route 53 範例](#)
- [使用 Tools for PowerShell V4 的 Amazon S3 範例](#)
- [使用 Tools for PowerShell V4 的 S3 Glacier 範例](#)
- [使用 Tools for PowerShell V4 的 Security Hub 範例](#)
- [使用 Tools for PowerShell V4 的 Amazon SES 範例](#)
- [使用 Tools for PowerShell V4 的 Amazon SES API v2 範例 V4](#)
- [使用 Tools for PowerShell V4 的 Amazon SNS 範例](#)
- [使用 Tools for PowerShell V4 的 Amazon SQS 範例](#)
- [AWS STS 使用 Tools for PowerShell V4 的範例](#)
- [支援 使用 Tools for PowerShell V4 的範例](#)
- [使用 Tools for PowerShell V4 的 Systems Manager 範例](#)

- [使用 Tools for PowerShell V4 的 Amazon Translate 範例](#)
- [AWS WAFV2 使用 Tools for PowerShell V4 的範例](#)
- [使用 Tools for PowerShell V4 的 WorkSpaces 範例](#)

使用 Tools for PowerShell V4 的 ACM 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 ACM 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Get-ACMCertificate

以下程式碼範例顯示如何使用 Get-ACMCertificate。

PowerShell V4 的工具

範例 1：此範例示範如何使用憑證的 ARN 傳回憑證及其鏈結。

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

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetCertificate](#)。

Get-ACMCertificateDetail

以下程式碼範例顯示如何使用 Get-ACMCertificateDetail。

PowerShell V4 的工具

範例 1：傳回指定憑證的詳細資訊。

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

輸出：

```
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}
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeCertificate](#)。

Get-ACMCertificateList

以下程式碼範例顯示如何使用 Get-ACMCertificateList。

PowerShell V4 的工具

範例 1：擷取所有憑證 ARNs 的清單，以及每個憑證的網域名稱。Cmdlet 會自動分頁以擷取所有 ARNs。若要手動控制分頁，請使用 -MaxItem 參數來控制每個服務呼叫傳回多少憑證 ARNs 並使用 -NextToken 參數來指出每個呼叫的起點。

```
Get-ACMCertificateList
```

輸出：

```

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

```

範例 2：擷取憑證狀態符合所提供狀態的所有憑證 ARNs 清單。

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

範例 3：此範例會傳回 us-east-1 區域中具有 RSA_2048 金鑰類型，以及 CODE_SIGNING 擴充金鑰用量或用途的所有憑證清單。您可以在 ListCertificates Filters API 參考主題中找到這些篩選參數的值：https://docs.aws.amazon.com/acm/latest/APIReference/API_Filters.html。

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

輸出：

```

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

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListCertificates](#)。

New-ACMCertificate

以下程式碼範例顯示如何使用 New-ACMCertificate。

PowerShell V4 的工具

範例 1：建立新的憑證。服務會傳回新憑證的 ARN。

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

輸出：

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

範例 2：建立新的憑證。服務會傳回新憑證的 ARN。

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

輸出：

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [RequestCertificate](#)。

Remove-ACMCertificate

以下程式碼範例顯示如何使用 Remove-ACMCertificate。

PowerShell V4 的工具

範例 1：刪除由提供的 ARN 識別的憑證和相關聯的私有金鑰。Cmdlet 會在繼續之前提示確認；新增 -強制切換來隱藏確認。

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

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteCertificate](#)。

Send-ACMValidationEmail

以下程式碼範例顯示如何使用 Send-ACMValidationEmail。

PowerShell V4 的工具

範例 1：請求傳送電子郵件以驗證 'www.example.com' 的網域擁有權。如果您殼層的 \$ConfirmPreference 設定為 'Medium' 或更低，Cmdlet 會在繼續之前提示確認。新增 - 強制切換以隱藏確認提示。

```
$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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [ResendValidationEmail](#)。

使用 Tools for PowerShell V4 的 Application Auto Scaling 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 Application Auto Scaling 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執程式碼的指示。

主題

- [動作](#)

動作

Add-AASScalableTarget

以下程式碼範例顯示如何使用 Add-AASScalableTarget。

PowerShell V4 的工具

範例 1：此 cmdlet 會註冊或更新可擴展的目標。可擴展的目標是 Application Auto Scaling 可以向
外擴展和向內擴展的資源。

```
Add-AASScalableTarget -ServiceNamespace AppStream -ResourceId fleet/MyFleet -  
ScalableDimension appstream:fleet:DesiredCapacity -MinCapacity 2 -MaxCapacity 10
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [RegisterScalableTarget](#)。

Get-AASScalableTarget

以下程式碼範例顯示如何使用 Get-AASScalableTarget。

PowerShell V4 的工具

範例 1：此範例將提供指定命名空間中 Application Autoscaling 可擴展目標的相關資訊。

```
Get-AASScalableTarget -ServiceNamespace "AppStream"
```

輸出：

```
CreationTime      : 11/7/2019 2:30:03 AM  
MaxCapacity       : 5  
MinCapacity       : 1  
ResourceId        : fleet/Test  
RoleARN           : arn:aws:iam::012345678912:role/aws-  
service-role/appstream.application-autoscaling.amazonaws.com/  
AWSServiceRoleForApplicationAutoScaling_AppStreamFleet  
ScalableDimension : appstream:fleet:DesiredCapacity  
ServiceNamespace  : appstream  
SuspendedState    : Amazon.ApplicationAutoScaling.Model.SuspendedState
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeScalableTargets](#)。

Get-AASScalingActivity

以下程式碼範例顯示如何使用 Get-AASScalingActivity。

PowerShell V4 的工具

範例 1：提供過去六週內指定命名空間中擴展活動的描述性資訊。

```
Get-AASScalingActivity -ServiceNamespace AppStream
```

輸出：

```
ActivityId      : 2827409f-b639-4cdb-a957-8055d5d07434
Cause           : monitor alarm Appstream2-MyFleet-default-scale-in Alarm in state
                 ALARM triggered policy default-scale-in
Description     : Setting desired capacity to 2.
Details        :
EndTime         : 12/14/2019 11:32:49 AM
ResourceId      : fleet/MyFleet
ScalableDimension : appstream:fleet:DesiredCapacity
ServiceNamespace : appstream
StartTime       : 12/14/2019 11:32:14 AM
StatusCode      : Successful
StatusMessage   : Successfully set desired capacity to 2. Change successfully
                 fulfilled by appstream.
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeScalingActivities](#)。

Get-AASScalingPolicy

以下程式碼範例顯示如何使用 Get-AASScalingPolicy。

PowerShell V4 的工具

範例 1：此 cmdlet 說明指定服務命名空間的 Application Auto Scaling 擴展政策。

```
Get-AASScalingPolicy -ServiceNamespace AppStream
```

輸出：

```

Alarms                                : {Appstream2-LabFleet-default-scale-out-
Alarm}
CreationTime                          : 9/3/2019 2:48:15 AM
PolicyARN                             : arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:5659b069-b5cd-4af1-9f7f-3e956d36233e:resource/
appstream/fleet/LabFleet:
                                     policyName/default-scale-out
PolicyName                            : default-scale-out
PolicyType                            : StepScaling
ResourceId                            : fleet/LabFleet
ScalableDimension                     : appstream:fleet:DesiredCapacity
ServiceNamespace                      : appstream
StepScalingPolicyConfiguration        :
    Amazon.ApplicationAutoScaling.Model.StepScalingPolicyConfiguration
TargetTrackingScalingPolicyConfiguration :

Alarms                                : {Appstream2-LabFleet-default-scale-in-
Alarm}
CreationTime                          : 9/3/2019 2:48:15 AM
PolicyARN                             : arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:5659b069-b5cd-4af1-9f7f-3e956d36233e:resource/
appstream/fleet/LabFleet:
                                     policyName/default-scale-in
PolicyName                            : default-scale-in
PolicyType                            : StepScaling
ResourceId                            : fleet/LabFleet
ScalableDimension                     : appstream:fleet:DesiredCapacity
ServiceNamespace                      : appstream
StepScalingPolicyConfiguration        :
    Amazon.ApplicationAutoScaling.Model.StepScalingPolicyConfiguration
TargetTrackingScalingPolicyConfiguration :

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeScalingPolicies](#)。

Get-AASScheduledAction

以下程式碼範例顯示如何使用 Get-AASScheduledAction。

PowerShell V4 的工具

範例 1：此 cmdlet 會列出為 Auto Scaling 群組排程且尚未執行或尚未到達結束時間的動作。

```
Get-AASScheduledAction -ServiceNamespace AppStream
```

輸出：

```
CreationTime      : 12/22/2019 9:25:52 AM
EndTime          : 1/1/0001 12:00:00 AM
ResourceId       : fleet/MyFleet
ScalableDimension : appstream:fleet:DesiredCapacity
ScalableTargetAction : Amazon.ApplicationAutoScaling.Model.ScalableTargetAction
Schedule         : cron(0 0 8 ? * MON-FRI *)
ScheduledActionARN : arn:aws:autoscaling:us-
west-2:012345678912:scheduledAction:4897ca24-3caa-4bf1-8484-851a089b243c:resource/
appstream/fleet/MyFleet:scheduledActionName
                  /WeekDaysFleetScaling
ScheduledActionName : WeekDaysFleetScaling
ServiceNamespace  : appstream
StartTime         : 1/1/0001 12:00:00 AM
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeScheduledActions](#)。

Remove-AASScalableTarget

以下程式碼範例顯示如何使用 Remove-AASScalableTarget。

PowerShell V4 的工具

範例 1：此 cmdlet 會取消註冊 Application Auto Scaling 可擴展目標。取消註冊可擴展目標會刪除與其相關聯的擴展政策。

```
Remove-AASScalableTarget -ResourceId fleet/MyFleet -ScalableDimension
appstream:fleet:DesiredCapacity -ServiceNamespace AppStream
```

輸出：

```
Confirm
Are you sure you want to perform this action?
```

```
Performing the operation "Remove-AASScalableTarget (DeregisterScalableTarget)" on
target "fleet/MyFleet".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeregisterScalableTarget](#)。

Remove-AASScalingPolicy

以下程式碼範例顯示如何使用 Remove-AASScalingPolicy。

PowerShell V4 的工具

範例 1：此 cmdlet 會刪除 Application Auto Scaling 可擴展目標的指定擴展政策。

```
Remove-AASScalingPolicy -ServiceNamespace AppStream -PolicyName "default-scale-out"
-ResourceId fleet/Test -ScalableDimension appstream:fleet:DesiredCapacity
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteScalingPolicy](#)。

Remove-AASScheduledAction

以下程式碼範例顯示如何使用 Remove-AASScheduledAction。

PowerShell V4 的工具

範例 1：此 cmdlet 會刪除 Application Auto Scaling 可擴展目標的指定排程動作。

```
Remove-AASScheduledAction -ServiceNamespace AppStream -ScheduledActionName
WeekDaysFleetScaling -ResourceId fleet/MyFleet -ScalableDimension
appstream:fleet:DesiredCapacity
```

輸出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-AASScheduledAction (DeleteScheduledAction)" on
target "WeekDaysFleetScaling".
```

```
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"): Y
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteScheduledAction](#)。

Set-AASScalingPolicy

以下程式碼範例顯示如何使用 Set-AASScalingPolicy。

PowerShell V4 的工具

範例 1：此 cmdlet 會建立或更新 Application Auto Scaling 可擴展目標的政策。每個可擴展的目標都由服務命名空間、資源 ID 和可擴展維度識別。

```
Set-AASScalingPolicy -ServiceNamespace AppStream -PolicyName ASFleetScaleInPolicy
-PolicyType StepScaling -ResourceId fleet/MyFleet -ScalableDimension
appstream:fleet:DesiredCapacity -StepScalingPolicyConfiguration_AdjustmentType
ChangeInCapacity -StepScalingPolicyConfiguration_Cooldown 360
-StepScalingPolicyConfiguration_MetricAggregationType Average -
StepScalingPolicyConfiguration_StepAdjustments @{ScalingAdjustment = -1;
MetricIntervalUpperBound = 0}
```

輸出：

```
Alarms      PolicyARN
-----
{}          arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:4897ca24-3caa-4bf1-8484-851a089b243c:resource/
appstream/fleet/MyFleet:policyName/ASFleetScaleInPolicy
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [PutScalingPolicy](#)。

Set-AASScheduledAction

以下程式碼範例顯示如何使用 Set-AASScheduledAction。

PowerShell V4 的工具

範例 1：此 cmdlet 會建立或更新 Application Auto Scaling 可擴展目標的排程動作。每個可擴展的目標都由服務命名空間、資源 ID 和可擴展維度識別。

```
Set-AASScheduledAction -ServiceNamespace AppStream -ResourceId fleet/MyFleet -Schedule "cron(0 0 8 ? * MON-FRI *)" -ScalableDimension appstream:fleet:DesiredCapacity -ScheduledActionName WeekDaysFleetScaling -ScalableTargetAction_MinCapacity 5 -ScalableTargetAction_MaxCapacity 10
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [PutScheduledAction](#)。

使用 Tools for PowerShell V4 的 AppStream 2.0 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 AppStream 2.0 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Add-APSResourceTag

以下程式碼範例顯示如何使用 Add-APSResourceTag。

PowerShell V4 的工具

範例 1：此範例會將資源標籤新增至 AppStream 資源

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

輸出：

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [TagResource](#)。

Copy-APSIImage

以下程式碼範例顯示如何使用 Copy-APSIImage。

PowerShell V4 的工具

範例 1：此範例會將映像複製到其他區域

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

輸出：

```
TestImageCopy
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CopyImage](#)。

Disable-APSUser

以下程式碼範例顯示如何使用 Disable-APSUser。

PowerShell V4 的工具

範例 1：此範例會停用 USERPOOL 中的 使用者

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DisableUser](#)。

Enable-APUser

以下程式碼範例顯示如何使用 Enable-APUser。

PowerShell V4 的工具

範例 1：此範例會在 USERPOOL 中啟用已停用的使用者

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [EnableUser](#)。

Get-APSAssociatedFleetList

以下程式碼範例顯示如何使用 Get-APSAssociatedFleetList。

PowerShell V4 的工具

範例 1：此範例顯示與堆疊相關聯的機群

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

輸出：

```
PowershellFleet
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListAssociatedFleets](#)。

Get-APSAssociatedStackList

以下程式碼範例顯示如何使用 Get-APSAssociatedStackList。

PowerShell V4 的工具

範例 1：此範例顯示與機群相關聯的堆疊

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

輸出：

```
PowershellStack
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [ListAssociatedStacks](#)。

Get-APSDirectoryConfigList

以下程式碼範例顯示如何使用 Get-APSDirectoryConfigList。

PowerShell V4 的工具

範例 1：此範例顯示 AppStream 中建立的目錄組態

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

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeDirectoryConfigs](#)。

Get-APSFleetList

以下程式碼範例顯示如何使用 Get-APSFleetList。

PowerShell V4 的工具

範例 1：此範例顯示機群的詳細資訊

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

輸出：

```
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

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeFleets](#)。

Get-APSIImageBuilderList

以下程式碼範例顯示如何使用 Get-APSIImageBuilderList。

PowerShell V4 的工具

範例 1：此範例顯示 ImageBuilder 的詳細資訊

```
Get-APSIImageBuilderList -Name TestImage
```

輸出：

```

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

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeImageBuilders](#)。

Get-APSIImageList

以下程式碼範例顯示如何使用 Get-APSIImageList。

PowerShell V4 的工具

範例 1：此範例顯示私有 AppStream 映像

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

輸出：

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeImages](#)。

Get-APSIImagePermission

以下程式碼範例顯示如何使用 Get-APSIImagePermission。

PowerShell V4 的工具

範例 1：此範例顯示共用 AppStream 映像上的映像許可

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

輸出：

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeImagePermissions](#)。

Get-APSSessionList

以下程式碼範例顯示如何使用 Get-APSSessionList。

PowerShell V4 的工具

範例 1：此範例顯示機群的工作階段清單

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

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeSessions](#)。

Get-APSSStackList

以下程式碼範例顯示如何使用 Get-APSSStackList。

PowerShell V4 的工具

範例 1：此範例顯示 AppStream Stack 的清單

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

輸出：

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeStacks](#)。

Get-APSTagsForResourceList

以下程式碼範例顯示如何使用 Get-APSTagsForResourceList。

PowerShell V4 的工具

範例 1：此範例顯示 AppStream 資源上的標籤

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

輸出：

Key	Value

```

---      -----
StackState Test

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListTagsForResource](#)。

Get-APSUsageReportSubscription

以下程式碼範例顯示如何使用 Get-APSUsageReportSubscription。

PowerShell V4 的工具

範例 1：此範例顯示 AppStreamUsageReport 組態詳細資訊

```
Get-APSUsageReportSubscription
```

輸出：

```

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

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeUsageReportSubscriptions](#)。

Get-APSUser

以下程式碼範例顯示如何使用 Get-APSUser。

PowerShell V4 的工具

範例 1：此範例顯示已啟用狀態的使用者清單

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

輸出：

```

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

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeUsers](#)。

Get-APSTUserStackAssociation

以下程式碼範例顯示如何使用 Get-APSTUserStackAssociation。

PowerShell V4 的工具

範例 1：此範例顯示指派給堆疊的使用者清單

```
Get-APSTUserStackAssociation -StackName PowershellStack
```

輸出：

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeUserStackAssociations](#)。

New-APSDirectoryConfig

以下程式碼範例顯示如何使用 New-APSDirectoryConfig。

PowerShell V4 的工具

範例 1：此範例會在 AppStream 中建立目錄組態

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

輸出：

```

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

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [CreateDirectoryConfig](#)。

New-APSFleet

以下程式碼範例顯示如何使用 New-APSFleet。

PowerShell V4 的工具

範例 1：此範例會建立新的 AppStream 機群

```

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

```

輸出：

```

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

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateFleet](#)。

New-APSIImageBuilder

以下程式碼範例顯示如何使用 New-APSIImageBuilder。

PowerShell V4 的工具

範例 1：此範例會在 AppStream 中建立映像建置器

```

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

```

輸出：

```

AccessEndpoints          : {}
AppstreamAgentVersion    : 12-16-2019
Arn                      : arn:aws:appstream:us-west-2:123456789012:image-
builder/TestIB
CreatedTime              : 12/27/2019 11:39:24 AM
Description              :
DisplayName              : TestIB
DomainJoinInfo           :
EnableDefaultInternetAccess : True
IamRoleArn               :
ImageArn                 : arn:aws:appstream:us-west-2::image/AppStream-
WinServer2012R2-12-12-2019
ImageBuilderErrors       : {}
InstanceType             : stream.standard.medium
Name                     : TestIB

```

```
NetworkAccessConfiguration :
Platform                   : WINDOWS
State                     : PENDING
StateChangeReason         :
VpcConfig                 : Amazon.AppStream.Model.VpcConfig
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [CreateImageBuilder](#)。

New-APSIImageBuilderStreamingURL

以下程式碼範例顯示如何使用 New-APSIImageBuilderStreamingURL。

PowerShell V4 的工具

範例 1：此範例會建立有效期為 2 小時的 ImageBuilder 串流 URL

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

輸出：

```
Expires                StreamingURL
-----                -
12/27/2019 1:49:13 PM https://appstream2.us-west-2.aws.amazon.com/authenticate?
parameters=eyJ0eXB1IjoiQURNSU4iLCJleHBpcmVzIjoiMTU3NzQ1NDU1MyIsImF3c0FjY291bnRJZCI6IjM5MzQwM
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateImageBuilderStreamingURL](#)。

New-APSSStack

以下程式碼範例顯示如何使用 New-APSSStack。

PowerShell V4 的工具

範例 1：此範例建立新的 AppStream 堆疊

```
New-APSSStack -Name TestStack -DisplayName TestStack -ApplicationSettings_Enabled
$True -ApplicationSettings_SettingsGroup TestStack -Region us-west-2
```


輸出：

```
AccessEndpoints      : {}
ApplicationSettings  : Amazon.AppStream.Model.ApplicationSettingsResponse
Arn                  : arn:aws:appstream:us-west-2:123456789012:stack/TestStack
CreatedTime          : 12/27/2019 12:34:19 PM
Description          :
DisplayName           : TestStack
EmbedHostDomains     : {}
FeedbackURL          :
Name                 : TestStack
RedirectURL           :
StackErrors          : {}
StorageConnectors    : {}
UserSettings         : {Amazon.AppStream.Model.UserSetting,
                        Amazon.AppStream.Model.UserSetting, Amazon.AppStream.Model.UserSetting,
                        Amazon.AppStream.Model.UserSetting}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateStack](#)。

New-APSSStreamingURL

以下程式碼範例顯示如何使用 New-APSSStreamingURL。

PowerShell V4 的工具

範例 1：此範例會建立 Stack 的串流 URL

```
New-APSSStreamingURL -StackName SessionScriptTest -FleetName SessionScriptNew -UserId
TestUser
```

輸出：

```
Expires              StreamingURL
-----
12/27/2019 12:43:37 PM https://appstream2.us-east-1.aws.amazon.com/authenticate?
parameters=eyJ0eXB1IjoiRU5EX1VTRVIiLCJleHBpcmVzIjoiMTU3NzQ1MDYxNyIsImF3c0FjY291bnRJZCI6IjM5M...
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateStreamingURL](#)。

New-APSUsageReportSubscription

以下程式碼範例顯示如何使用 New-APSUsageReportSubscription。

PowerShell V4 的工具

範例 1：此範例啟用 AppStream 用量報告

```
New-APSUsageReportSubscription
```

輸出：

S3BucketName	Schedule
-----	-----
appstream-logs-us-east-1-123456789012-sik2hnxe	DAILY

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateUsageReportSubscription](#)。

New-APSUser

以下程式碼範例顯示如何使用 New-APSUser。

PowerShell V4 的工具

範例 1：此範例會在 USERPOOL 中建立使用者

```
New-APSUser -UserName Test@lab.com -AuthenticationType USERPOOL -FirstName 'kt' -  
LastName 'aws' -Select ^UserName
```

輸出：

```
Test@lab.com
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [CreateUser](#)。

Register-APSFleet

以下程式碼範例顯示如何使用 Register-APSFleet。

PowerShell V4 的工具

範例 1：此範例向堆疊註冊機群

```
Register-APSFleet -StackName TestStack -FleetName TestFleet -Region us-west-2
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [AssociateFleet](#)。

Register-APSUserStackBatch

以下程式碼範例顯示如何使用 Register-APSUserStackBatch。

PowerShell V4 的工具

範例 1：此範例會將堆疊指派給 USERPOOL 中的使用者

```
Register-APSUserStackBatch -UserStackAssociation  
@{AuthenticationType="USERPOOL";SendEmailNotification=  
$False;StackName="PowershellStack";UserName="TestUser1@lab.com"}
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [BatchAssociateUserStack](#)。

Remove-APSDirectoryConfig

以下程式碼範例顯示如何使用 Remove-APSDirectoryConfig。

PowerShell V4 的工具

範例 1：此範例會移除 AppStream Directory 組態

```
Remove-APSDirectoryConfig -DirectoryName contoso.com
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteDirectoryConfig](#)。

Remove-APSFleet

以下程式碼範例顯示如何使用 Remove-APSFleet。

PowerShell V4 的工具

範例 1：此範例移除會刪除 AppStream 機群

```
Remove-APSFleet -Name TestFleet -Region us-west-2
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteFleet](#)。

Remove-APSIImage

以下程式碼範例顯示如何使用 Remove-APSIImage。

PowerShell V4 的工具

範例 1：此範例會刪除映像

```
Remove-APSIImage -Name TestImage -Region us-west-2
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteImage](#)。

Remove-APSIImageBuilder

以下程式碼範例顯示如何使用 Remove-APSIImageBuilder。

PowerShell V4 的工具

範例 1：此範例會刪除 ImageBuilder

```
Remove-APSIImageBuilder -Name TestIB -Region us-west-2
```

輸出：

```
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

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteImageBuilder](#)。

Remove-APSIImagePermission

以下程式碼範例顯示如何使用 Remove-APSIImagePermission。

PowerShell V4 的工具

範例 1：此範例會移除映像的許可

```
Remove-APSIImagePermission -Name Powershell -SharedAccountId 123456789012
```

輸出：

```

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

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteImagePermissions](#)。

Remove-APSResourceTag

以下程式碼範例顯示如何使用 Remove-APSResourceTag。

PowerShell V4 的工具

範例 1：此範例會從 AppStream 資源移除資源標籤

```
Remove-APSResourceTag -ResourceArn arn:aws:appstream:us-east-1:123456789012:stack/SessionScriptTest -TagKey StackState
```

輸出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSResourceTag (UntagResource)" on target
"arn:aws:appstream:us-east-1:123456789012:stack/SessionScriptTest".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): A
```

- 如需 API 詳細資訊，請參閱《Cmdlet 參考》中的 [UntagResource](#) (V4)。AWS Tools for PowerShell

Remove-APSStack

以下程式碼範例顯示如何使用 Remove-APSStack。

PowerShell V4 的工具

範例 1：此範例會刪除堆疊

```
Remove-APSStack -Name TestStack -Region us-west-2
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱《Cmdlet 參考》中的 [DeleteStack](#) (V4)。AWS Tools for PowerShell

Remove-APSUsageReportSubscription

以下程式碼範例顯示如何使用 Remove-APSUsageReportSubscription。

PowerShell V4 的工具

範例 1：此範例停用 AppStream 用量報告訂閱

```
Remove-APSUsageReportSubscription
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteUsageReportSubscription](#)。

Remove-APSUser

以下程式碼範例顯示如何使用 Remove-APSUser。

PowerShell V4 的工具

範例 1：此範例會從 USERPOOL 刪除使用者

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

輸出：


```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteUser](#)。

Revoke-APSSession

以下程式碼範例顯示如何使用 Revoke-APSSession。

PowerShell V4 的工具

範例 1：此範例會撤銷 AppStream 機群的工作階段

```
Revoke-APSSession -SessionId 6cd2f9a3-f948-4aa1-8014-8a7dcde14877
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ExpireSession](#)。

Start-APSFleet

以下程式碼範例顯示如何使用 Start-APSFleet。

PowerShell V4 的工具

範例 1：此範例會啟動機群

```
Start-APSFleet -Name PowershellFleet
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [StartFleet](#)。

Start-APSIImageBuilder

以下程式碼範例顯示如何使用 Start-APSIImageBuilder。

PowerShell V4 的工具

範例 1：此範例會啟動 ImageBuilder

```
Start-APSIImageBuilder -Name TestImage
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [StartImageBuilder](#)。

Stop-APSFleet

以下程式碼範例顯示如何使用 Stop-APSFleet。

PowerShell V4 的工具

範例 1：此範例會停止機群

```
Stop-APSFleet -Name PowershellFleet
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [StopFleet](#)。

Stop-APSIImageBuilder

以下程式碼範例顯示如何使用 Stop-APSIImageBuilder。

PowerShell V4 的工具

範例 1：此範例會停止 ImageBuilder

```
Stop-APSIImageBuilder -Name TestImage
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [StopImageBuilder](#)。

Unregister-APSFleet

以下程式碼範例顯示如何使用 Unregister-APSFleet。

PowerShell V4 的工具

範例 1：此範例會從堆疊取消註冊機群

```
Unregister-APSFleet -StackName TestStack -FleetName TestFleet -Region us-west-2
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DisassociateFleet](#)。

Unregister-APSUserStackBatch

以下程式碼範例顯示如何使用 Unregister-APSUserStackBatch。

PowerShell V4 的工具

範例 1：此範例會從指派的 Stack 移除使用者

```
Unregister-APSUserStackBatch -UserStackAssociation
@{AuthenticationType="USERPOOL";SendEmailNotification=
$False;StackName="PowershellStack";UserName="TestUser1@lab.com"}
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [BatchDisassociateUserStack](#)。

Update-APSDirectoryConfig

以下程式碼範例顯示如何使用 Update-APSDirectoryConfig。

PowerShell V4 的工具

範例 1：此範例會更新在 AppStream 中建立的目錄組態

```
Update-APSDirectoryConfig -ServiceAccountCredentials_AccountName contoso
\ServiceAccount -ServiceAccountCredentials_AccountPassword MyPass@1$@#
-DirectoryName contoso.com -OrganizationalUnitDistinguishedName
"OU=AppStreamNew,OU=Contoso,DC=Contoso,DC=com"
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [UpdateDirectoryConfig](#)。

Update-APSFleet

以下程式碼範例顯示如何使用 Update-APSFleet。

PowerShell V4 的工具

範例 1：此範例會更新機群的屬性

```
Update-APSFleet -Name PowershellFleet -EnableDefaultInternetAccess $True -
DisconnectTimeoutInSeconds 950
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [UpdateFleet](#)。

Update-APSIImagePermission

以下程式碼範例顯示如何使用 Update-APSIImagePermission。

PowerShell V4 的工具

範例 1：此範例與其他帳戶共用 AppStream 映像

```
Update-APSIImagePermission -Name Powershell -SharedAccountId 123456789012 -
ImagePermissions_AllowFleet $True -ImagePermissions_AllowImageBuilder $True
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [UpdateImagePermissions](#)。

Update-APSStack

以下程式碼範例顯示如何使用 Update-APSStack。

PowerShell V4 的工具

範例 1：此範例更新（啟用）堆疊上的應用程式設定持續性和主資料夾

```
Update-APSStack -Name PowershellStack -ApplicationSettings_Enabled $True
-ApplicationSettings_SettingsGroup PowershellStack -StorageConnector
@{ConnectorType="HOMEFOLDERS"}
```

輸出：

```
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}
```

- 如需 API 詳細資訊，請參閱《Cmdlet 參考》中的 [UpdateStack](#) (V4)。AWS Tools for PowerShell

使用 Tools for PowerShell V4 的 Aurora 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 Aurora 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Get-RDSOrderableDBInstanceOption

以下程式碼範例顯示如何使用 Get-RDSOrderableDBInstanceOption。

PowerShell V4 的工具

範例 1：此範例列出支援中特定資料庫執行個體類別的資料庫引擎版本 AWS 區域。

```
$params = @{
  Engine = 'aurora-postgresql'
  DBInstanceClass = 'db.r5.large'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

範例 2：此範例列出 中特定資料庫引擎版本支援的資料庫執行個體類別 AWS 區域。

```
$params = @{  
    Engine = 'aurora-postgresql'  
    EngineVersion = '13.6'  
    Region = 'us-east-1'  
}  
Get-RDSOrderableDBInstanceOption @params
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeOrderableDBInstanceOptions](#)。

使用 Tools for PowerShell V4 的 Auto Scaling 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 Auto Scaling 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Add-ASLoadBalancer

以下程式碼範例顯示如何使用 Add-ASLoadBalancer。

PowerShell V4 的工具

範例 1：此範例會將指定的負載平衡器連接到指定的 Auto Scaling 群組。

```
Add-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [AttachLoadBalancers](#)。

Complete-ASLifecycleAction

以下程式碼範例顯示如何使用 Complete-ASLifecycleAction。

PowerShell V4 的工具

範例 1：此範例會完成指定的生命週期動作。

```
Complete-ASLifecycleAction -LifecycleHookName myLifecycleHook -  
AutoScalingGroupName my-asg -LifecycleActionResult CONTINUE -LifecycleActionToken  
bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CompleteLifecycleAction](#)。

Disable-ASMetricsCollection

以下程式碼範例顯示如何使用 Disable-ASMetricsCollection。

PowerShell V4 的工具

範例 1：此範例會停用監控指定 Auto Scaling 群組的指定指標。

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg -Metric @("GroupMinSize",  
"GroupMaxSize")
```

範例 2：此範例會停用監控指定 Auto Scaling 群組的所有指標。

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DisableMetricsCollection](#)。

Dismount-ASInstance

以下程式碼範例顯示如何使用 Dismount-ASInstance。

PowerShell V4 的工具

範例 1：此範例會將指定的執行個體與指定的 Auto Scaling 群組分離，並減少所需的容量，讓 Auto Scaling 不會啟動替代執行個體。

```
Dismount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

輸出：

```
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        :
```

範例 2：此範例會將指定的執行個體與指定的 Auto Scaling 群組分離，而不會減少所需的容量。Auto Scaling 會啟動替換執行個體。

```
Dismount-ASInstance -InstanceId i-7bf746a2 -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

輸出：

```
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        :
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DetachInstances](#)。

Dismount-ASLoadBalancer

以下程式碼範例顯示如何使用 Dismount-ASLoadBalancer。

PowerShell V4 的工具

範例 1：此範例會將指定的負載平衡器與指定的 Auto Scaling 群組分離。

```
Dismount-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DetachLoadBalancers](#)。

Enable-ASMetricsCollection

以下程式碼範例顯示如何使用 Enable-ASMetricsCollection。

PowerShell V4 的工具

範例 1：此範例可監控指定 Auto Scaling 群組的指定指標。

```
Enable-ASMetricsCollection -Metric @("GroupMinSize", "GroupMaxSize") -  
AutoScalingGroupName my-asg -Granularity 1Minute
```

範例 2：此範例可監控指定 Auto Scaling 群組的所有指標。

```
Enable-ASMetricsCollection -AutoScalingGroupName my-asg -Granularity 1Minute
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [EnableMetricsCollection](#)。

Enter-ASStandby

以下程式碼範例顯示如何使用 Enter-ASStandby。

PowerShell V4 的工具

範例 1：此範例會將指定的執行個體置於待命模式，並減少所需的容量，讓 Auto Scaling 不會啟動替代執行個體。

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -  
ShouldDecrementDesiredCapacity $true
```

輸出：

```
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         :
```

範例 2：此範例會將指定的執行個體置於待命模式，而不會減少所需的容量。Auto Scaling 會啟動替換執行個體。

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -  
ShouldDecrementDesiredCapacity $false
```

輸出：

```
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  :

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [EnterStandby](#)。

Exit-ASStandby

以下程式碼範例顯示如何使用 Exit-ASStandby。

PowerShell V4 的工具

範例 1：此範例會將指定的執行個體移出待命模式。

```
Exit-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

輸出：

```

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  :

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ExitStandby](#)。

Get-ASAccountLimit

以下程式碼範例顯示如何使用 Get-ASAccountLimit。

PowerShell V4 的工具

範例 1：此範例說明您 AWS 帳戶的 Auto Scaling 資源限制。

```
Get-ASAccountLimit
```

輸出：

```
MaxNumberOfAutoScalingGroups      : 20
MaxNumberOfLaunchConfigurations    : 100
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeAccountLimits](#)。

Get-ASAdjustmentType

以下程式碼範例顯示如何使用 Get-ASAdjustmentType。

PowerShell V4 的工具

範例 1：此範例說明 Auto Scaling 支援的調整類型。

```
Get-ASAdjustmentType
```

輸出：

```
Type
----
ChangeInCapacity
ExactCapacity
PercentChangeInCapacity
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeAdjustmentTypes](#)。

Get-ASAutoScalingGroup

以下程式碼範例顯示如何使用 Get-ASAutoScalingGroup。

PowerShell V4 的工具

範例 1：此範例列出 Auto Scaling 群組的名稱。

```
Get-ASAutoScalingGroup | format-table -property AutoScalingGroupName
```

輸出：

```
AutoScalingGroupName
-----
my-asg-1
my-asg-2
my-asg-3
my-asg-4
my-asg-5
my-asg-6
```

範例 2：此範例說明指定的 Auto Scaling 群組。

```
Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1
```

輸出：

```
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
```

範例 3：此範例說明指定的兩個 Auto Scaling 群組。

```
Get-ASAutoScalingGroup -AutoScalingGroupName @"my-asg-1", "my-asg-2")
```

範例 4：此範例說明指定 Auto Scaling 群組的 Auto Scaling 執行個體。

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1).Instances
```

範例 5：此範例說明所有 Auto Scaling 群組。

```
Get-ASAutoScalingGroup
```

範例 6：此範例說明指定 Auto Scaling 群組的 LaunchTemplate。此範例假設「執行個體購買選項」設定為「Adhere to launch template」。如果此選項設定為「合併購買選項和執行個體類型」，可以使用「MixedInstancesPolicy.LaunchTemplate」屬性存取 LaunchTemplate。MixedInstancesPolicy.LaunchTemplate"

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-ag-1).LaunchTemplate
```

輸出：

LaunchTemplateId	LaunchTemplateName	Version
-----	-----	-----
lt-06095fd619cb40371	test-launch-template	\$Default

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeAutoScalingGroups](#)。

Get-ASAutoScalingInstance

以下程式碼範例顯示如何使用 Get-ASAutoScalingInstance。

PowerShell V4 的工具

範例 1：此範例列出 Auto Scaling 執行個體 IDs。


```
Get-ASAutoScalingInstance | format-table -property InstanceId
```

輸出：

```
InstanceId
-----
i-12345678
i-87654321
i-abcd1234
```

範例 2：此範例說明指定的 Auto Scaling 執行個體。

```
Get-ASAutoScalingInstance -InstanceId i-12345678
```

輸出：

```
AutoScalingGroupName    : my-asg
AvailabilityZone         : us-west-2b
HealthStatus             : HEALTHY
InstanceId               : i-12345678
LaunchConfigurationName : my-lc
LifecycleState           : InService
```

範例 3：此範例說明指定的兩個 Auto Scaling 執行個體。

```
Get-ASAutoScalingInstance -InstanceId @"i-12345678", "i-87654321"
```

範例 4：此範例說明指定 Auto Scaling 群組的 Auto Scaling 執行個體。

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg).Instances | Get-
ASAutoScalingInstance
```

範例 5：此範例說明所有 Auto Scaling 執行個體。

```
Get-ASAutoScalingInstance
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeAutoScalingInstances](#)。

Get-ASAutoScalingNotificationType

以下程式碼範例顯示如何使用 Get-ASAutoScalingNotificationType。

PowerShell V4 的工具

範例 1：此範例列出 Auto Scaling 支援的通知類型。

```
Get-ASAutoScalingNotificationType
```

輸出：

```
autoscaling:EC2_INSTANCE_LAUNCH
autoscaling:EC2_INSTANCE_LAUNCH_ERROR
autoscaling:EC2_INSTANCE_TERMINATE
autoscaling:EC2_INSTANCE_TERMINATE_ERROR
autoscaling:TEST_NOTIFICATION
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeAutoScalingNotificationTypes](#)。

Get-ASLaunchConfiguration

以下程式碼範例顯示如何使用 Get-ASLaunchConfiguration。

PowerShell V4 的工具

範例 1：此範例列出啟動組態的名稱。

```
Get-ASLaunchConfiguration | format-table -property LaunchConfigurationName
```

輸出：

```
LaunchConfigurationName
-----
my-lc-1
my-lc-2
my-lc-3
my-lc-4
my-lc-5
```

範例 2：此範例說明指定的啟動組態。

```
Get-ASLaunchConfiguration -LaunchConfigurationName my-lc-1
```

輸出：

```
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                     :
```

範例 3：此範例說明指定的兩個啟動組態。

```
Get-ASLaunchConfiguration -LaunchConfigurationName @("my-lc-1", "my-lc-2")
```

範例 4：此範例說明所有啟動組態。

```
Get-ASLaunchConfiguration
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeLaunchConfigurations](#)。

Get-ASLifecycleHook

以下程式碼範例顯示如何使用 Get-ASLifecycleHook。

PowerShell V4 的工具

範例 1：此範例說明指定的生命週期關聯。

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName myLifecycleHook
```

輸出：

```
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
```

範例 2：此範例說明指定 Auto Scaling 群組的所有生命週期關聯。

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg
```

範例 3：此範例說明所有 Auto Scaling 群組的所有生命週期關聯。

```
Get-ASLifecycleHook
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeLifecycleHooks](#)。

Get-ASLifecycleHookType

以下程式碼範例顯示如何使用 Get-ASLifecycleHookType。

PowerShell V4 的工具

範例 1：此範例列出 Auto Scaling 支援的生命週期掛鉤類型。

```
Get-ASLifecycleHookType
```

輸出：

```
autoscaling:EC2_INSTANCE_LAUNCHING  
auto-scaling:EC2_INSTANCE_TERMINATING
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeLifecycleHookTypes](#)。

Get-ASLoadBalancer

以下程式碼範例顯示如何使用 Get-ASLoadBalancer。

PowerShell V4 的工具

範例 1：此範例說明指定 Auto Scaling 群組的負載平衡器。

```
Get-ASLoadBalancer -AutoScalingGroupName my-asg
```

輸出：

LoadBalancerName	State
-----	----
my-lb	Added

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeLoadBalancers](#)。

Get-ASMetricCollectionType

以下程式碼範例顯示如何使用 Get-ASMetricCollectionType。

PowerShell V4 的工具

範例 1：此範例列出 Auto Scaling 支援的指標集合類型。

```
(Get-ASMetricCollectionType).Metrics
```

輸出：

```
Metric
-----
GroupMinSize
GroupMaxSize
GroupDesiredCapacity
GroupInServiceInstances
GroupPendingInstances
GroupTerminatingInstances
GroupStandbyInstances
GroupTotalInstances
```

範例 2：此範例列出對應的精細程度。

```
(Get-ASMetricCollectionType).Granularities
```

輸出：

```
Granularity
-----
1Minute
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeMetricCollectionTypes](#)。

Get-ASNotificationConfiguration

以下程式碼範例顯示如何使用 Get-ASNotificationConfiguration。

PowerShell V4 的工具

範例 1：此範例說明與指定 Auto Scaling 群組相關聯的通知動作。

```
Get-ASNotificationConfiguration -AutoScalingGroupName my-asg | format-list
```

輸出：

```
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

```

範例 2：此範例說明與所有 Auto Scaling 群組相關聯的通知動作。

```
Get-ASNotificationConfiguration
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeNotificationConfigurations](#)。

Get-ASPolicy

以下程式碼範例顯示如何使用 Get-ASPolicy。

PowerShell V4 的工具

範例 1：此範例說明指定 Auto Scaling 群組的所有政策。

```
Get-ASPolicy -AutoScalingGroupName my-asg
```

輸出：

```

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      : {}

```

範例 2：此範例說明指定 Auto Scaling 群組的指定政策。

```
Get-ASPolicy -AutoScalingGroupName my-asg -PolicyName @("myScaleOutPolicy",
"myScaleInPolicy")
```

範例 3：此範例說明所有 Auto Scaling 群組的所有政策。

```
Get-ASPolicy
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribePolicies](#)。

Get-ASScalingActivity

以下程式碼範例顯示如何使用 Get-ASScalingActivity。

PowerShell V4 的工具

範例 1：此範例說明指定 Auto Scaling 群組過去六週的擴展活動。

```
Get-ASScalingActivity -AutoScalingGroupName my-asg
```

輸出：

```
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 between
                      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        :
```

範例 2：此範例說明指定的擴展活動。

```
Get-ASScalingActivity -ActivityId "063308ae-aa22-4a9b-94f4-9fae4EXAMPLE"
```

範例 3：此範例說明所有 Auto Scaling 群組過去六週的擴展活動。

```
Get-ASScalingActivity
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeScalingActivities](#)。

Get-ASScalingProcessType

以下程式碼範例顯示如何使用 Get-ASScalingProcessType。

PowerShell V4 的工具

範例 1：此範例列出 Auto Scaling 支援的程序類型。

```
Get-ASScalingProcessType
```

輸出：

```

ProcessName
-----
AZRebalance
AddToLoadBalancer
AlarmNotification
HealthCheck
Launch
ReplaceUnhealthy
ScheduledActions
Terminate

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeScalingProcessTypes](#)。

Get-ASScheduledAction

以下程式碼範例顯示如何使用 Get-ASScheduledAction。

PowerShell V4 的工具

範例 1：此範例說明指定 Auto Scaling 群組的排程擴展動作。

```
Get-ASScheduledAction -AutoScalingGroupName my-asg
```

輸出：

```

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

```

範例 2：此範例說明指定的排程擴展動作。

```
Get-ASScheduledAction -ScheduledActionName @("myScheduledScaleOut",
"myScheduledScaleIn")
```

範例 3：此範例描述從指定時間開始的排程擴展動作。

```
Get-ASScheduledAction -StartTime "2015-12-01T08:00:00Z"
```

範例 4：此範例描述以指定時間結束的排程擴展動作。

```
Get-ASScheduledAction -EndTime "2015-12-30T08:00:00Z"
```

範例 5：此範例說明所有 Auto Scaling 群組的排程擴展動作。

```
Get-ASScheduledAction
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeScheduledActions](#)。

Get-ASTag

以下程式碼範例顯示如何使用 Get-ASTag。

PowerShell V4 的工具

範例 1：此範例說明鍵值為 'myTag' 或 'myTag2' 的標籤。篩選條件名稱的可能值為 'auto-scaling-group'、'key'、'value' 和 'propagate-at-launch'。此範例使用的語法需要 PowerShell 版本 3 或更新版本。

```
Get-ASTag -Filter @( @{ Name="key"; Values=@("myTag", "myTag2") } )
```

輸出：

```
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
```

範例 2：使用 PowerShell 第 2 版時，您必須使用 `New-Object` 來建立篩選條件參數的篩選條件。

```
$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 )
```

範例 3：此範例說明所有 Auto Scaling 群組的所有標籤。

```
Get-ASTag
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeTags](#)。

Get-ASTerminationPolicyType

以下程式碼範例顯示如何使用 `Get-ASTerminationPolicyType`。

PowerShell V4 的工具

範例 1：此範例列出 Auto Scaling 支援的終止政策。

```
Get-ASTerminationPolicyType
```

輸出：

```
ClosestToNextInstanceHour
Default
NewestInstance
```

```
OldestInstance  
OldestLaunchConfiguration
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeTerminationPolicyTypes](#)。

Mount-ASInstance

以下程式碼範例顯示如何使用 Mount-ASInstance。

PowerShell V4 的工具

範例 1：此範例會將指定的執行個體連接至指定的 Auto Scaling 群組。Auto Scaling 會自動增加 Auto Scaling 群組所需的容量。

```
Mount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [AttachInstances](#)。

New-ASAutoScalingGroup

以下程式碼範例顯示如何使用 New-ASAutoScalingGroup。

PowerShell V4 的工具

範例 1：此範例會建立具有指定名稱和屬性的 Auto Scaling 群組。預設所需容量為最小大小。因此，此 Auto Scaling 群組會啟動兩個執行個體，其中一個位於指定的兩個可用區域中。

```
New-ASAutoScalingGroup -AutoScalingGroupName my-asg -LaunchConfigurationName my-lc -  
MinSize 2 -MaxSize 6 -AvailabilityZone @"us-west-2a", "us-west-2b")
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateAutoScalingGroup](#)。

New-ASLaunchConfiguration

以下程式碼範例顯示如何使用 New-ASLaunchConfiguration。

PowerShell V4 的工具

範例 1：此範例會建立名為 'my-lc' 的啟動組態。使用此啟動組態的 Auto Scaling 群組啟動的 EC2 執行個體會使用指定的執行個體類型、AMI、安全群組和 IAM 角色。

```
New-ASLaunchConfiguration -LaunchConfigurationName my-lc -InstanceType "m3.medium" -  
ImageId "ami-12345678" -SecurityGroup "sg-12345678" -IamInstanceProfile "myIamRole"
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateLaunchConfiguration](#)。

Remove-ASAutoScalingGroup

以下程式碼範例顯示如何使用 Remove-ASAutoScalingGroup。

PowerShell V4 的工具

範例 1：如果沒有執行中的執行個體，此範例會刪除指定的 Auto Scaling 群組。在操作繼續之前，系統會提示您進行確認。

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg
```

輸出：

```
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"):
```

範例 2：如果您指定強制參數，在操作繼續之前不會提示您進行確認。

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -Force
```

範例 3：此範例會刪除指定的 Auto Scaling 群組，並終止其包含的任何執行中執行個體。

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -ForceDelete $true -Force
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteAutoScalingGroup](#)。

Remove-ASLaunchConfiguration

以下程式碼範例顯示如何使用 Remove-ASLaunchConfiguration。

PowerShell V4 的工具

範例 1：如果未連接到 Auto Scaling 群組，此範例會刪除指定的啟動組態。在操作繼續之前，系統會提示您進行確認。

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc
```

輸出：

```
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"):
```

範例 2：如果您指定強制參數，在操作繼續之前不會提示您進行確認。

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc -Force
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteLaunchConfiguration](#)。

Remove-ASLifecycleHook

以下程式碼範例顯示如何使用 Remove-ASLifecycleHook。

PowerShell V4 的工具

範例 1：此範例會刪除指定 Auto Scaling 群組的指定生命週期關聯。在操作繼續之前，系統會提示您進行確認。

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName  
myLifecycleHook
```

輸出：

```
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"):
```

範例 2：如果您指定強制參數，在操作繼續之前，不會提示您進行確認。

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName  
myLifecycleHook -Force
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteLifecycleHook](#)。

Remove-ASNotificationConfiguration

以下程式碼範例顯示如何使用 Remove-ASNotificationConfiguration。

PowerShell V4 的工具

範例 1：此範例會刪除指定的通知動作。在操作繼續之前，系統會提示您進行確認。

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN  
"arn:aws:sns:us-west-2:123456789012:my-topic"
```

輸出：

```
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"):
```


範例 2：如果您指定強制參數，在操作繼續之前，不會提示您進行確認。

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN  
"arn:aws:sns:us-west-2:123456789012:my-topic" -Force
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteNotificationConfiguration](#)。

Remove-ASPolicy

以下程式碼範例顯示如何使用 Remove-ASPolicy。

PowerShell V4 的工具

範例 1：此範例會刪除指定 Auto Scaling 群組的指定政策。在操作繼續之前，系統會提示您進行確認。

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy
```

輸出：

```
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"):
```

範例 2：如果您指定強制參數，在操作繼續之前不會提示您進行確認。

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy -Force
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeletePolicy](#)。

Remove-ASScheduledAction

以下程式碼範例顯示如何使用 Remove-ASScheduledAction。

PowerShell V4 的工具

範例 1：此範例會刪除指定 Auto Scaling 群組的指定排程動作。在操作繼續之前，系統會提示您進行確認。

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction  
"myScheduledAction"
```

輸出：

```
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"):
```

範例 2：如果您指定強制參數，在操作繼續之前不會提示您進行確認。

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction  
"myScheduledAction" -Force
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteScheduledAction](#)。

Remove-ASTag

以下程式碼範例顯示如何使用 Remove-ASTag。

PowerShell V4 的工具

範例 1：此範例會從指定的 Auto Scaling 群組中移除指定的標籤。在操作繼續之前，系統會提示您進行確認。此範例使用的語法需要 PowerShell 版本 3 或更新版本。

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";  
Key="myTag" } )
```

輸出：

```
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"):
```

範例 2：如果您指定強制參數，在操作繼續之前不會提示您進行確認。

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
Key="myTag" } ) -Force
```

範例 3：使用 Powershell 第 2 版時，您必須使用 New-Object 來建立標籤參數的標籤。

```
$tag = New-Object Amazon.AutoScaling.Model.Tag
$tag.ResourceType = "auto-scaling-group"
$tag.ResourceId = "my-asg"
$tag.Key = "myTag"
Remove-ASTag -Tag $tag -Force
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteTags](#)。

Resume-ASProcess

以下程式碼範例顯示如何使用 Resume-ASProcess。

PowerShell V4 的工具

範例 1：此範例會繼續指定 Auto Scaling 群組的指定 Auto Scaling 程序。

```
Resume-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"
```

範例 2：此範例會繼續指定 Auto Scaling 群組的所有暫停 Auto Scaling 程序。

```
Resume-ASProcess -AutoScalingGroupName my-asg
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ResumeProcesses](#)。

Set-ASDesiredCapacity

以下程式碼範例顯示如何使用 Set-ASDesiredCapacity。

PowerShell V4 的工具

範例 1：此範例會設定指定 Auto Scaling 群組的大小。

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2
```

範例 2：此範例會設定指定 Auto Scaling 群組的大小，並等待冷卻時間完成，然後再擴展到新的大小。

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2 -HonorCooldown $true
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [SetDesiredCapacity](#)。

Set-ASInstanceHealth

以下程式碼範例顯示如何使用 Set-ASInstanceHealth。

PowerShell V4 的工具

範例 1：此範例會將指定執行個體的狀態設定為 'Unhealthy'，使其停止服務。Auto Scaling 會終止並取代執行個體。

```
Set-ASInstanceHealth -HealthStatus Unhealthy -InstanceId i-93633f9b
```

範例 2：此範例會將指定執行個體的狀態設定為 'Healthy'，使其保持服務狀態。不會遵守 Auto Scaling 群組的任何運作狀態檢查寬限期。

```
Set-ASInstanceHealth -HealthStatus Healthy -InstanceId i-93633f9b -ShouldRespectGracePeriod $false
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [SetInstanceHealth](#)。

Set-ASInstanceProtection

以下程式碼範例顯示如何使用 Set-ASInstanceProtection。

PowerShell V4 的工具

範例 1：此範例會啟用指定執行個體的執行個體保護。

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -  
ProtectedFromScaleIn $true
```

範例 2：此範例會停用指定執行個體的執行個體保護。

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -  
ProtectedFromScaleIn $false
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [SetInstanceProtection](#)。

Set-ASTag

以下程式碼範例顯示如何使用 Set-ASTag。

PowerShell V4 的工具

範例 1：此範例會將單一標籤新增至指定的 Auto Scaling 群組。標籤鍵為 'myTag'，標籤值為 'myTagValue'。Auto Scaling 會將此標籤傳播到 Auto Scaling 群組啟動的後續 EC2 執行個體。此範例使用的語法需要 PowerShell 版本 3 或更新版本。

```
Set-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";  
Key="myTag"; Value="myTagValue"; PropagateAtLaunch=$true} )
```

範例 2：使用 PowerShell 第 2 版時，您必須使用 New-Object 來建立標籤參數的標籤。

```
$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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateOrUpdateTags](#)。

Start-ASPolicy

以下程式碼範例顯示如何使用 Start-ASPolicy。

PowerShell V4 的工具

範例 1：此範例會為指定的 Auto Scaling 群組執行指定的政策。

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy"
```

範例 2：此範例會在等待冷卻時間完成之後，為指定的 Auto Scaling 群組執行指定的政策。

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy" -  
HonorCooldown $true
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ExecutePolicy](#)。

Stop-ASInstanceInAutoScalingGroup

以下程式碼範例顯示如何使用 Stop-ASInstanceInAutoScalingGroup。

PowerShell V4 的工具

範例 1：此範例會終止指定的執行個體，並減少其 Auto Scaling 群組所需的容量，讓 Auto Scaling 不會啟動替代執行個體。

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -  
ShouldDecrementDesiredCapacity $true
```

輸出：

```
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  :

```

範例 2：此範例會終止指定的執行個體，而不會減少其 Auto Scaling 群組所需的容量。Auto Scaling 會啟動替換執行個體。

```

Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $false

```

輸出：

```

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  :

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [TerminateInstanceInAutoScalingGroup](#)。

Suspend-ASProcess

以下程式碼範例顯示如何使用 Suspend-ASProcess。

PowerShell V4 的工具

範例 1：此範例會暫停指定 Auto Scaling 群組的指定 Auto Scaling 程序。

```

Suspend-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"

```

範例 2：此範例會暫停指定 Auto Scaling 群組的所有 Auto Scaling 程序。

```
Suspend-ASProcess -AutoScalingGroupName my-asg
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [SuspendProcesses](#)。

Update-ASAutoScalingGroup

以下程式碼範例顯示如何使用 Update-ASAutoScalingGroup。

PowerShell V4 的工具

範例 1：此範例會更新指定 Auto Scaling 群組的大小下限和上限。

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -MaxSize 5 -MinSize 1
```

範例 2：此範例會更新指定 Auto Scaling 群組的預設冷卻時間。

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -DefaultCooldown 10
```

範例 3：此範例會更新指定 Auto Scaling 群組的可用區域。

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -AvailabilityZone @("us-west-2a", "us-west-2b")
```

範例 4：此範例會更新指定的 Auto Scaling 群組，以使用 Elastic Load Balancing 運作狀態檢查。

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -HealthCheckType ELB -  
HealthCheckGracePeriod 60
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [UpdateAutoScalingGroup](#)。

Write-ASLifecycleActionHeartbeat

以下程式碼範例顯示如何使用 Write-ASLifecycleActionHeartbeat。

PowerShell V4 的工具

範例 1：此範例會記錄指定生命週期動作的活動訊號。這會使執行個體保持待定狀態，直到您完成自訂動作為止。


```
Write-ASLifecycleActionHeartbeat -AutoScalingGroupName my-asg -LifecycleHookName  
myLifecycleHook -LifecycleActionToken bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [RecordLifecycleActionHeartbeat](#)。

Write-ASLifecycleHook

以下程式碼範例顯示如何使用 Write-ASLifecycleHook。

PowerShell V4 的工具

範例 1：此範例會將指定的生命週期掛鉤新增至指定的 Auto Scaling 群組。

```
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"
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [PutLifecycleHook](#)。

Write-ASNotificationConfiguration

以下程式碼範例顯示如何使用 Write-ASNotificationConfiguration。

PowerShell V4 的工具

範例 1：此範例會設定指定的 Auto Scaling 群組，以在啟動 EC2 執行個體時傳送通知至指定的 SNS 主題。

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType  
"autoscaling:EC2_INSTANCE_LAUNCH" -TopicARN "arn:aws:sns:us-west-2:123456789012:my-  
topic"
```

範例 2：此範例會將指定的 Auto Scaling 群組設定為在啟動或終止 EC2 執行個體時，將通知傳送至指定的 SNS 主題。

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType
@("autoscaling:EC2_INSTANCE_LAUNCH", "autoscaling:EC2_INSTANCE_TERMINATE") -
TopicARN "arn:aws:sns:us-west-2:123456789012:my-topic"
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [PutNotificationConfiguration](#)。

Write-ASScalingPolicy

以下程式碼範例顯示如何使用 Write-ASScalingPolicy。

PowerShell V4 的工具

範例 1：此範例會將指定的政策新增至指定的 Auto Scaling 群組。指定的調整類型決定如何解譯 ScalingAdjustment 參數。使用「ChangeInCapacity」時，正值會將容量增加指定的執行個體數量，負值則會將容量減少指定的執行個體數量。

```
Write-ASScalingPolicy -AutoScalingGroupName my-asg -AdjustmentType
"ChangeInCapacity" -PolicyName "myScaleInPolicy" -ScalingAdjustment -1
```

輸出：

```
arn:aws:autoscaling:us-west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-
e1d769fc24ef:autoScalingGroupName/my-asg
:policyName/myScaleInPolicy
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [PutScalingPolicy](#)。

Write-ASScheduledUpdateGroupAction

以下程式碼範例顯示如何使用 Write-ASScheduledUpdateGroupAction。

PowerShell V4 的工具

範例 1：此範例會建立或更新一次性排程動作，以在指定的開始時間變更所需的容量。

```
Write-ASScheduledUpdateGroupAction -AutoScalingGroupName my-asg -ScheduledActionName
"myScheduledAction" -StartTime "2015-12-01T00:00:00Z" -DesiredCapacity 10
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [PutScheduledUpdateGroupAction](#)。

AWS Budgets 使用 Tools for PowerShell V4 的範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 來執行動作和實作常見案例 AWS Budgets。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

New-BGTBudget

以下程式碼範例顯示如何使用 New-BGTBudget。

PowerShell V4 的工具

範例 1：使用電子郵件通知建立具有指定預算和時間限制的新預算。

```
$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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateBudget](#)。

AWS Cloud9 使用 Tools for PowerShell V4 的範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 來執行動作和實作常見案例 AWS Cloud9。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Get-C9EnvironmentData

以下程式碼範例顯示如何使用 Get-C9EnvironmentData。

PowerShell V4 的工具

範例 1：此範例會取得指定 AWS Cloud9 開發環境的相關資訊。

```
Get-C9EnvironmentData -EnvironmentId
685f892f431b45c2b28cb69eadcdb0EX,1980b80e5f584920801c09086667f0EX
```

輸出：

```

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

```

範例 2：此範例會取得指定 AWS Cloud9 開發環境生命週期狀態的相關資訊。

```
(Get-C9EnvironmentData -EnvironmentId 685f892f431b45c2b28cb69eadcdb0EX).Lifecycle
```

輸出：

```

FailureResource Reason Status
-----
                        CREATED

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeEnvironments](#)。

Get-C9EnvironmentList

以下程式碼範例顯示如何使用 Get-C9EnvironmentList。

PowerShell V4 的工具

範例 1：此範例會取得 available AWS Cloud9 開發環境識別符的清單。

```
Get-C9EnvironmentList
```

輸出：

```
685f892f431b45c2b28cb69eadcdb0EX
1980b80e5f584920801c09086667f0EX
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListEnvironments](#)。

Get-C9EnvironmentMembershipList

以下程式碼範例顯示如何使用 Get-C9EnvironmentMembershipList。

PowerShell V4 的工具

範例 1：此範例會取得指定 AWS Cloud9 開發環境的環境成員相關資訊。

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeeaeaa8a04bfde8cEX
```

輸出：

```
EnvironmentId : ffd88420d4824eeeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-write
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA602FMJWCXHEX

EnvironmentId : ffd88420d4824eeeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3L0R0M0UXTBSU6EX
```

範例 2：此範例會取得指定 AWS Cloud9 開發環境擁有者的相關資訊。

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeeaeaa8a04bfde8cEX -
Permission owner
```

輸出：

```
EnvironmentId : ffd88420d4824eeeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
```

```
Permissions      : owner
UserArn          : arn:aws:iam::123456789012:user/MyDemoUser
UserId           : AIDAJ3LOROMOUXTBSU6EX
```

範例 3：此範例會取得多個 AWS Cloud9 開發環境之指定環境成員的相關資訊。

```
Get-C9EnvironmentMembershipList -UserArn arn:aws:iam::123456789012:user/MyDemoUser
```

輸出：

```
EnvironmentId : ffd88420d4824eeeeaeaa8a04bfde8cEX
LastAccess    : 1/17/2018 7:48:14 PM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX

EnvironmentId : 1980b80e5f584920801c09086667f0EX
LastAccess    : 1/16/2018 11:21:24 PM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeEnvironmentMemberships](#)。

Get-C9EnvironmentStatus

以下程式碼範例顯示如何使用 Get-C9EnvironmentStatus。

PowerShell V4 的工具

範例 1：此範例會取得指定 AWS Cloud9 開發環境的狀態資訊。

```
Get-C9EnvironmentStatus -EnvironmentId 349c86d4579e4e7298d500ff57a6b2EX
```

輸出：

```
Message                Status
-----
Environment is ready to use ready
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeEnvironmentStatus](#)。

New-C9EnvironmentEC2

以下程式碼範例顯示如何使用 New-C9EnvironmentEC2。

PowerShell V4 的工具

範例 1：此範例會使用指定的設定建立 an AWS Cloud9 開發環境、啟動 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體，然後從執行個體連線至環境。

```
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
```

輸出：

```
ffd88420d4824eeeeaaa8a04bfde8cEX
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateEnvironmentEc2](#)。

New-C9EnvironmentMembership

以下程式碼範例顯示如何使用 New-C9EnvironmentMembership。

PowerShell V4 的工具

範例 1：此範例會將指定的環境成員新增至指定的 AWS Cloud9 開發環境。

```
New-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/AnotherDemoUser
-EnvironmentId ffd88420d4824eeeeaaa8a04bfde8cEX -Permission read-write
```

輸出：

```
EnvironmentId : ffd88420d4824eeeeaaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-write
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
```



```
UserId : AIDAJ3BA602FMJWCWXHEX
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateEnvironmentMembership](#)。

Remove-C9Environment

以下程式碼範例顯示如何使用 Remove-C9Environment。

PowerShell V4 的工具

範例 1：此範例會刪除指定的 AWS Cloud9 開發環境。如果 Amazon EC2 執行個體連線到環境，也會終止執行個體。

```
Remove-C9Environment -EnvironmentId ffd88420d4824eeeeaea8a04bfde8cEX
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteEnvironment](#)。

Remove-C9EnvironmentMembership

以下程式碼範例顯示如何使用 Remove-C9EnvironmentMembership。

PowerShell V4 的工具

範例 1：此範例會從指定的 AWS Cloud9 開發環境刪除指定的環境成員。

```
Remove-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/  
AnotherDemoUser -EnvironmentId ffd88420d4824eeeeaea8a04bfde8cEX
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteEnvironmentMembership](#)。

Update-C9Environment

以下程式碼範例顯示如何使用 Update-C9Environment。

PowerShell V4 的工具

範例 1：此範例會變更指定之 existing AWS Cloud9 開發環境的指定設定。

```
Update-C9Environment -EnvironmentId ffd88420d4824eeeeaea8a04bfde8cEX -Description  
"My changed demonstration development environment." -Name my-changed-demo-env
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UpdateEnvironment](#)。

Update-C9EnvironmentMembership

以下程式碼範例顯示如何使用 Update-C9EnvironmentMembership。

PowerShell V4 的工具

範例 1：此範例會變更指定 AWS Cloud9 開發環境之指定現有環境成員的設定。

```
Update-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/  
AnotherDemoUser -EnvironmentId ffd88420d4824eeeeaea8a04bfde8cEX -Permission read-  
only
```

輸出：

```
EnvironmentId : ffd88420d4824eeeeaea8a04bfde8cEX  
LastAccess    : 1/1/0001 12:00:00 AM  
Permissions   : read-only  
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser  
UserId        : AIDAJ3BA602FMJWCWXHEX
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [UpdateEnvironmentMembership](#)。

AWS CloudFormation 使用 Tools for PowerShell V4 的範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 來執行動作和實作常見案例 AWS CloudFormation。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Get-CFNStack

以下程式碼範例顯示如何使用 Get-CFNStack。

PowerShell V4 的工具

範例 1：傳回描述所有使用者堆疊的堆疊執行個體集合。

```
Get-CFNStack
```

範例 2：傳回描述指定堆疊的堆疊執行個體

```
Get-CFNStack -StackName "myStack"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeStacks](#)。

Get-CFNStackEvent

以下程式碼範例顯示如何使用 Get-CFNStackEvent。

PowerShell V4 的工具

範例 1：傳回指定堆疊的所有堆疊相關事件。

```
Get-CFNStackEvent -StackName "myStack"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeStackEvents](#)。

Get-CFNStackResource

以下程式碼範例顯示如何使用 Get-CFNStackResource。

PowerShell V4 的工具

範例 1：傳回透過邏輯 ID "MyDBInstance" 在與指定堆疊相關聯的範本中識別的資源描述。

```
Get-CFNStackResource -StackName "myStack" -LogicalResourceId "MyDBInstance"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeStackResource](#)。

Get-CFNStackResourceList

以下程式碼範例顯示如何使用 Get-CFNStackResourceList。

PowerShell V4 的工具

範例 1：傳回最多 100 個與指定堆疊相關聯的資源 AWS 描述。若要取得與堆疊相關聯的所有資源詳細資訊，請使用 Get-CFNStackResourceSummary，這也支援手動分頁結果。

```
Get-CFNStackResourceList -StackName "myStack"
```

範例 2：傳回透過邏輯 ID "Ec2Instance" 在與指定堆疊相關聯的範本中識別的 Amazon EC2 執行個體描述。Ec2Instance

```
Get-CFNStackResourceList -StackName "myStack" -LogicalResourceId "Ec2Instance"
```

範例 3：傳回最多 100 個與堆疊相關聯的資源描述，其中包含由執行個體 ID "i-123456" 識別的 Amazon EC2 執行個體。若要取得與堆疊相關聯的所有資源詳細資訊，請使用 Get-CFNStackResourceSummary，這也支援手動分頁結果。

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456"
```

範例 4：傳回由堆疊範本中的邏輯 ID "Ec2Instance" 所識別的 Amazon EC2 執行個體描述。Ec2Instance 堆疊是使用其包含的資源的實體資源 ID 來識別，在此情況下，也會是執行個體 ID 為 "i-123456" 的 Amazon EC2 執行個體。根據範本內容，也可以使用不同的實體資源來識別堆疊，例如 Amazon S3 儲存貯體。

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456" -LogicalResourceId "Ec2Instance"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeStackResources](#)。

Get-CFNStackResourceSummary

以下程式碼範例顯示如何使用 Get-CFNStackResourceSummary。

PowerShell V4 的工具

範例 1：傳回與指定堆疊相關聯的所有資源的描述。

```
Get-CFNStackResourceSummary -StackName "myStack"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListStackResources](#)。

Get-CFNStackSummary

以下程式碼範例顯示如何使用 Get-CFNStackSummary。

PowerShell V4 的工具

範例 1：傳回所有堆疊的摘要資訊。

```
Get-CFNStackSummary
```

範例 2：傳回目前正在建立之所有堆疊的摘要資訊。

```
Get-CFNStackSummary -StackStatusFilter "CREATE_IN_PROGRESS"
```

範例 3：傳回目前正在建立或更新之所有堆疊的摘要資訊。

```
Get-CFNStackSummary -StackStatusFilter @("CREATE_IN_PROGRESS", "UPDATE_IN_PROGRESS")
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListStacks](#)。

Get-CFNTemplate

以下程式碼範例顯示如何使用 Get-CFNTemplate。

PowerShell V4 的工具

範例 1：傳回與指定堆疊相關聯的範本。

```
Get-CFNTemplate -StackName "myStack"
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetTemplate](#)。

Measure-CFNTemplateCost

以下程式碼範例顯示如何使用 Measure-CFNTemplateCost。

PowerShell V4 的工具

範例 1：傳回 AWS 簡易每月計算器 URL，其中包含查詢字串，描述執行範本所需的資源。範本是從指定的 Amazon S3 URL 取得，並套用單一自訂參數。參數也可以使用 'Key' 和 'Value' 來指定，而不是 'ParameterKey' 和 'ParameterValue'。

```
Measure-CFNTemplateCost -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                        -Region us-west-1 `
                        -Parameter @{ ParameterKey="KeyName";
ParameterValue="myKeyPairName" }
```

範例 2：傳回 AWS 簡易每月計算器 URL，其中包含查詢字串，說明執行範本所需的資源。範本會從提供的內容剖析，並套用自訂參數（此範例假設範本內容已宣告兩個參數：'KeyName' 和 'InstanceType'）。自訂參數也可以使用 'Key' 和 'Value' 來指定，而不是 'ParameterKey' 和 'ParameterValue'。

```
Measure-CFNTemplateCost -TemplateBody "{TEMPLATE CONTENT HERE}" `
                        -Parameter @( @{ ParameterKey="KeyName";
ParameterValue="myKeyPairName" }, `
                                     @{ ParameterKey="InstanceType";
ParameterValue="m1.large" })
```

範例 3：使用 New-Object 建置一組範本參數，並傳回具有查詢字串的 AWS 簡易每月計算器 URL，該字串說明執行範本所需的資源。範本會從提供的內容中剖析，並使用自訂參數（此範例假設範本內容已宣告兩個參數：'KeyName' 和 '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 )
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [EstimateTemplateCost](#)。

New-CFNStack

以下程式碼範例顯示如何使用 New-CFNStack。

PowerShell V4 的工具

範例 1：建立具有指定名稱的新堆疊。範本是以自訂參數 ('PK1' 和 'PK2' 代表範本內容中宣告的參數名稱，'PV1' 和 'PV2' 代表這些參數的值。自訂參數也可以使用 'Key' 和 'Value' 來指定，而不是 'ParameterKey' 和 'ParameterValue'。如果堆疊的建立失敗，將不會復原。

```
New-CFNStack -StackName "myStack" `
    -TemplateBody "{TEMPLATE CONTENT HERE}" `
    -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
    @{ ParameterKey="PK2"; ParameterValue="PV2" } ) `
    -DisableRollback $true
```

範例 2：建立具有指定名稱的新堆疊。範本是以自訂參數 ('PK1' 和 'PK2' 代表範本內容中宣告的參數名稱，'PV1' 和 'PV2' 代表這些參數的值。自訂參數也可以使用 'Key' 和 'Value' 來指定，而不是 'ParameterKey' 和 'ParameterValue'。如果堆疊的建立失敗，則會復原。

```
$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"
```

範例 3：建立具有指定名稱的新堆疊。範本是從具有自訂參數的 Amazon S3 URL 取得 ('PK1' 代表範本內容中宣告的參數名稱，'PV1' 代表參數的值。自訂參數也可以使用 'Key' 和 'Value' 來指定，而不是 'ParameterKey' 和 'ParameterValue'。如果建立堆疊失敗，則會復原（與指定 -DisableRollback \$false 相同）。

```
New-CFNStack -StackName "myStack" `
              -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
              -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

範例 4：建立具有指定名稱的新堆疊。範本是從具有自訂參數的 Amazon S3 URL 取得 ('PK1' 代表範本內容中宣告的參數名稱，'PV1' 代表參數的值。自訂參數也可以使用 'Key' 和 'Value' 來指定，而不是 'ParameterKey' 和 'ParameterValue'。如果建立堆疊失敗，則會復原（與指定 -DisableRollback \$false 相同）。指定的通知 AENs 將收到已發佈的堆疊相關事件。

```
New-CFNStack -StackName "myStack" `
              -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
              -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" } `
              -NotificationARN @( "arn1", "arn2" )
```

- 如需 API 詳細資訊，請參閱《Cmdlet 參考》中的 [CreateStack](#) (V4)。AWS Tools for PowerShell

Remove-CFNStack

以下程式碼範例顯示如何使用 Remove-CFNStack。

PowerShell V4 的工具

範例 1：刪除指定的堆疊。

```
Remove-CFNStack -StackName "myStack"
```


- 如需 API 詳細資訊，請參閱《Cmdlet 參考》中的 [DeleteStack](#) (V4)。AWS Tools for PowerShell

Resume-CFNUpdateRollback

以下程式碼範例顯示如何使用 Resume-CFNUpdateRollback。

PowerShell V4 的工具

範例 1：繼續復原具名堆疊，其應處於「UPDATE_ROLLBACK_FAILED」狀態。如果繼續轉返成功，堆疊將進入狀態 'UPDATE_ROLLBACK_COMPLETE'。

```
Resume-CFNUpdateRollback -StackName "myStack"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ContinueUpdateRollback](#)。

Stop-CFNUpdateStack

以下程式碼範例顯示如何使用 Stop-CFNUpdateStack。

PowerShell V4 的工具

範例 1：取消指定堆疊的更新。

```
Stop-CFNUpdateStack -StackName "myStack"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CancelUpdateStack](#)。

Test-CFNStack

以下程式碼範例顯示如何使用 Test-CFNStack。

PowerShell V4 的工具

範例 1：測試堆疊是否已達到其中一個狀態
UPDATE_ROLLBACK_COMPLETE、CREATE_COMPLETE、ROLLBACK_COMPLETE 或
UPDATE_COMPLETE。

```
Test-CFNStack -StackName MyStack
```

輸出：

```
False
```

範例 2：測試堆疊是否已達到 UPDATE_COMPLETE 或 UPDATE_ROLLBACK_COMPLETE 狀態。

```
Test-CFNStack -StackName MyStack -Status UPDATE_COMPLETE,UPDATE_ROLLBACK_COMPLETE
```

輸出：

```
True
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [Test-CFNStack](#)。

Test-CFNTemplate

以下程式碼範例顯示如何使用 Test-CFNTemplate。

PowerShell V4 的工具

範例 1：驗證指定的範本內容。輸出會詳細說明範本的功能、描述和參數。

```
Test-CFNTemplate -TemplateBody "{TEMPLATE CONTENT HERE}"
```

範例 2：驗證透過 Amazon S3 URL 存取的指定範本。輸出會詳細說明範本的功能、描述和參數。

```
Test-CFNTemplate -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/  
templatefile.template
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [ValidateTemplate](#)。

Update-CFNStack

以下程式碼範例顯示如何使用 Update-CFNStack。

PowerShell V4 的工具

範例 1：使用指定的範本和自訂參數更新堆疊 'myStack'。'PK1' 代表範本中宣告的參數名稱，而 'PV1' 代表其值。自訂參數也可以使用 'Key' 和 'Value' 來指定，而不是 'ParameterKey' 和 'ParameterValue'。

```
Update-CFNStack -StackName "myStack" `
    -TemplateBody "{Template Content Here}" `
    -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

範例 2：使用指定的範本和自訂參數更新堆疊 'myStack'。'PK1' 和 'PK2' 代表範本中宣告的參數名稱，'PV1' 和 'PV2' 代表其請求的值。自訂參數也可以使用 'Key' 和 'Value' 來指定，而不是 'ParameterKey' 和 'ParameterValue'。

```
Update-CFNStack -StackName "myStack" `
    -TemplateBody "{Template Content Here}" `
    -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
    @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

範例 3：使用指定的範本和自訂參數更新堆疊 'myStack'。'PK1' 代表範本中宣告的參數名稱，而 'PV2' 代表其值。自訂參數也可以使用 'Key' 和 'Value' 來指定，而不是 'ParameterKey' 和 'ParameterValue'。

```
Update-CFNStack -StackName "myStack" -TemplateBody "{Template Content Here}" -
Parameters @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

範例 4：使用從 Amazon S3 取得的指定範本和自訂參數來更新堆疊 'myStack'。'PK1' 和 'PK2' 代表範本中宣告的參數名稱，'PV1' 和 'PV2' 代表其請求的值。自訂參數也可以使用 'Key' 和 'Value' 來指定，而不是 'ParameterKey' 和 'ParameterValue'。

```
Update-CFNStack -StackName "myStack" `
    -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
    -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
    @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

範例 5：更新堆疊 'myStack'，此範例假設包含 IAM 資源，其中包含從 Amazon S3 取得的指定範本和自訂參數。'PK1' 和 'PK2' 代表範本中宣告的參數名稱，'PV1' 和 'PV2' 代表其請求的值。

自訂參數也可以使用 'Key' 和 'Value' 來指定，而不是 'ParameterKey' 和 'ParameterValue'。包含 IAM 資源的堆疊需要您指定 -Capabilities "CAPABILITY_IAM" 參數，否則更新會失敗，並顯示 'InsufficientCapabilities' 錯誤。

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
@{ ParameterKey="PK2"; ParameterValue="PV2" } ) `
                -Capabilities "CAPABILITY_IAM"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UpdateStack](#)。

Wait-CFNStack

以下程式碼範例顯示如何使用 Wait-CFNStack。

PowerShell V4 的工具

範例 1：測試堆疊是否已達到其中一個狀態

UPDATE_ROLLBACK_COMPLETE、CREATE_COMPLETE、ROLLBACK_COMPLETE 或 UPDATE_COMPLETE。如果堆疊不在其中一個狀態，命令會休眠兩秒鐘，然後再次測試狀態。這會重複此動作，直到堆疊達到其中一個請求狀態，或經過 60 秒的預設逾時期間為止。如果超過逾時期間，則會擲回例外狀況。如果堆疊在逾時期間內達到其中一個請求的狀態，則會傳回管道。

```
$stack = Wait-CFNStack -StackName MyStack
```

範例 2：此範例會等待總共 5 分鐘 (300 秒)，讓堆疊達到其中一個指定的狀態。在此範例中，在逾時之前達到狀態，因此堆疊物件會傳回管道。

```
Wait-CFNStack -StackName MyStack -Timeout 300 -Status
CREATE_COMPLETE,ROLLBACK_COMPLETE
```

輸出：

```
Capabilities      : {CAPABILITY_IAM}
ChangeSetId       :
CreationTime      : 6/1/2017 9:29:33 AM
```

```

Description      : AWS CloudFormation Sample Template
ec2_instance_with_instance_profile: Create an EC2 instance with an associated
instance profile. **WARNING** This template creates one or more Amazon EC2
instances and an Amazon SQS queue. You will be billed for the
AWS resources used if you create a stack from this template.
DisableRollback  : False
LastUpdatedTime  : 1/1/0001 12:00:00 AM
NotificationARNs : {}
Outputs          : {}
Parameters       : {}
RoleARN          :
StackId          : arn:aws:cloudformation:us-west-2:123456789012:stack/
MyStack/7ea87b50-46e7-11e7-9c9b-503a90a9c4d1
StackName        : MyStack
StackStatus      : CREATE_COMPLETE
StackStatusReason :
Tags             : {}
TimeoutInMinutes : 0

```

範例 3：此範例顯示當堆疊在逾時期間（在此情況下為預設期間 60 秒）內未達到其中一個請求狀態時的錯誤輸出。

```
Wait-CFNStack -StackName MyStack -Status CREATE_COMPLETE,ROLLBACK_COMPLETE
```

輸出：

```

Wait-CFNStack : Timed out after 60 seconds waiting for CloudFormation
stack MyStack in region us-west-2 to reach one of state(s):
UPDATE_ROLLBACK_COMPLETE,CREATE_COMPLETE,ROLLBACK_COMPLETE,UPDATE_COMPLETE
At line:1 char:1
+ Wait-CFNStack -StackName MyStack -State CREATE_COMPLETE,ROLLBACK_COMPLETE
+ ~~~~~
+ CategoryInfo          : InvalidOperation:
(Amazon.PowerShe...tCFNStackCmdlet:WaitCFNStackCmdlet) [Wait-CFNStack],
InvalidOperationException
+ FullyQualifiedErrorId :
InvalidOperationException,Amazon.PowerShell.Cmdlets.CFN.WaitCFNStackCmdlet

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [Wait-CFNStack](#)。

使用 Tools for PowerShell V4 的 CloudFront 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 CloudFront 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Get-CFCloudFrontOriginAccessIdentity

以下程式碼範例顯示如何使用 Get-CFCloudFrontOriginAccessIdentity。

PowerShell V4 的工具

範例 1：此範例會傳回 -Id 參數指定的特定 Amazon CloudFront 原始存取身分。雖然不需要 -Id 參數，但如果您未指定它，則不會傳回任何結果。

```
Get-CFCloudFrontOriginAccessIdentity -Id E3XXXXXXXXXXRT
```

輸出：

```
CloudFrontOriginAccessIdentityConfig    Id
S3CanonicalUserId
-----
Amazon.CloudFront.Model.CloudFrontOr... E3XXXXXXXXXXRT
4b6e...
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetCloudFrontOriginAccessIdentity](#)。

Get-CFCloudFrontOriginAccessIdentityConfig

以下程式碼範例顯示如何使用 Get-CFCloudFrontOriginAccessIdentityConfig。

PowerShell V4 的工具

範例 1：此範例會傳回 -Id 參數所指定之單一 Amazon CloudFront 原始存取身分的組態資訊。如果未指定 -Id 參數，則會發生錯誤。

```
Get-CFCloudFrontOriginAccessIdentityConfig -Id E3XXXXXXXXXXRT
```

輸出：

CallerReference	Comment
-----	-----
mycallerreference: 2/1/2011 1:16:32 PM	Caller reference:
2/1/2011 1:16:32 PM	

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GetCloudFrontOriginAccessIdentityConfig](#)。

Get-CFCloudFrontOriginAccessIdentityList

以下程式碼範例顯示如何使用 Get-CFCloudFrontOriginAccessIdentityList。

PowerShell V4 的工具

範例 1：此範例會傳回 Amazon CloudFront 原始存取身分的清單。由於 -MaxItem 參數指定值 2，因此結果包含兩個身分。

```
Get-CFCloudFrontOriginAccessIdentityList -MaxItem 2
```

輸出：

```
IsTruncated : True
Items       : {E326XXXXXXXXXT, E1YWXXXXXXXX9B}
Marker      :
MaxItems    : 2
NextMarker  : E1YXXXXXXXXX9B
Quantity    : 2
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListCloudFrontOriginAccessIdentities](#)。

Get-CFDistribution

以下程式碼範例顯示如何使用 Get-CFDistribution。

PowerShell V4 的工具

範例 1：擷取特定分佈的資訊。

```
Get-CFDistribution -Id EXAMPLE0000ID
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetDistribution](#)。

Get-CFDistributionConfig

以下程式碼範例顯示如何使用 Get-CFDistributionConfig。

PowerShell V4 的工具

範例 1：擷取特定分佈的組態。

```
Get-CFDistributionConfig -Id EXAMPLE0000ID
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetDistributionConfig](#)。

Get-CFDistributionList

以下程式碼範例顯示如何使用 Get-CFDistributionList。

PowerShell V4 的工具

範例 1：傳回分佈。

```
Get-CFDistributionList
```


- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListDistributions](#)。

New-CFDistribution

以下程式碼範例顯示如何使用 New-CFDistribution。

PowerShell V4 的工具

範例 1：建立基本 CloudFront 分佈，設定記錄和快取。

```
$origin = New-Object Amazon.CloudFront.Model.Origin
$origin.DomainName = "amzn-s3-demo-bucket.s3.amazonaws.com"
$origin.Id = "UniqueOrigin1"
$origin.S3OriginConfig = New-Object Amazon.CloudFront.Model.S3OriginConfig
$origin.S3OriginConfig.OriginAccessIdentity = ""
New-CFDistribution `
    -DistributionConfig_Enabled $true `
    -DistributionConfig_Comment "Test distribution" `
    -Origins_Item $origin `
    -Origins_Quantity 1 `
    -Logging_Enabled $true `
    -Logging_IncludeCookie $true `
    -Logging_Bucket amzn-s3-demo-logging-bucket.s3.amazonaws.com `
    -Logging_Prefix "help/" `
    -DistributionConfig_CallerReference Client1 `
    -DistributionConfig_DefaultRootObject index.html `
    -DefaultCacheBehavior_TargetOriginId $origin.Id `
    -ForwardedValues_QueryString $true `
    -Cookies_Forward all `
    -WhitelistedNames_Quantity 0 `
    -TrustedSigners_Enabled $false `
    -TrustedSigners_Quantity 0 `
    -DefaultCacheBehavior_ViewerProtocolPolicy allow-all `
    -DefaultCacheBehavior_MinTTL 1000 `
    -DistributionConfig_PriceClass "PriceClass_All" `
    -CacheBehaviors_Quantity 0 `
    -Aliases_Quantity 0
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateDistribution](#)。

New-CFInvalidation

以下程式碼範例顯示如何使用 New-CFInvalidation。

PowerShell V4 的工具

範例 1：此範例會在 ID 為 EXAMPLENSTXAXE 的分佈上建立新的失效。CallerReference 是使用者選擇的唯一 ID；在此情況下，會使用代表 2019 年 5 月 15 日上午 9：00 的時間戳記。\$Paths 變數會存放使用者不希望作為分佈快取一部分的映像和媒體檔案的三個路徑。-Paths_Quantity 參數值是 -Paths_Item 參數中指定的路徑總數。

```
$Paths = "/images/*.gif", "/images/image1.jpg", "/videos/*.mp4"
New-CFInvalidation -DistributionId "EXAMPLENSTXAXE" -
InvalidationBatch_CallerReference 20190515090000 -Paths_Item $Paths -Paths_Quantity
3
```

輸出：

Invalidation	Location
-----	-----
Amazon.CloudFront.Model.Invalidation	https://cloudfront.amazonaws.com/2018-11-05/distribution/EXAMPLENSTXAXE/invalidation/EXAMPLE8N0K9H

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateInvalidation](#)。

New-CFSignedCookie

以下程式碼範例顯示如何使用 New-CFSignedCookie。

PowerShell V4 的工具

範例 1：使用標準政策為指定的資源建立簽章的 Cookie。Cookie 有效期為一年。

```
$params = @{
  "ResourceUri"="http://xyz.cloudfront.net/image1.jpeg"
  "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
  "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
  "ExpiresOn"=(Get-Date).AddYears(1)
```

```
}  
New-CFSignedCookie @params
```

輸出：

```
Expires  
-----  
[CloudFront-Expires, 1472227284]
```

範例 2：使用自訂政策為指定的資源建立簽章的 Cookie。Cookie 將在 24 小時內生效，並在一週後過期。

```
$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
```

輸出：

```
Policy  
-----  
[CloudFront-Policy, eyJTd...wIjo...
```

範例 3：使用自訂政策為指定的資源建立簽章的 Cookie。Cookie 將在 24 小時內生效，並在一週後過期。資源的存取僅限於指定的 IP 範圍。

```
$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
```

```
New-CFSignedCookie @params
```

輸出：

```
Policy
-----
[CloudFront-Policy, eyJTd...wIjo...
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [New-CFSignedCookie](#)。

New-CFSignedUrl

以下程式碼範例顯示如何使用 New-CFSignedUrl。

PowerShell V4 的工具

範例 1：使用標準政策建立指定資源的已簽署 URL。URL 有效期為一小時。包含已簽署 URL 的 System.Uri 物件會發送到管道。

```
$params = @{
  "ResourceUri"="https://cdn.example.com/index.html"
  "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
  "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
  "ExpiresOn"=(Get-Date).AddHours(1)
}
New-CFSignedUrl @params
```

範例 2：使用自訂政策建立指定資源的已簽署 URL。url 將在 24 小時內生效，並在一週後過期。

```
$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
```

範例 3：使用自訂政策建立指定資源的已簽署 URL。url 將在 24 小時內生效，並在一週後過期。資源的存取僅限於指定的 ip 範圍。

```
$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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [New-CFSignedUrl](#)。

使用 Tools for PowerShell V4 的 CloudTrail 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 CloudTrail 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Find-CTEvent

以下程式碼範例顯示如何使用 Find-CTEvent。

PowerShell V4 的工具

範例 1：傳回過去七天內發生的所有事件。根據預設，cmdlet 會自動進行多次呼叫以交付所有事件，並在服務指出沒有其他資料可用時結束。

```
Find-CTEvent
```

範例 2：傳回過去七天內發生的所有事件，指定不是目前 shell 預設值的區域。

```
Find-CTEvent -Region eu-central-1
```

範例 3：傳回與 RunInstances API 呼叫相關聯的所有事件。

```
Find-CTEvent -LookupAttribute @{ AttributeKey="EventName";  
    AttributeValue="RunInstances" }
```

範例 4：傳回前 5 個可用的事件。

```
Find-CTEvent -MaxResult 5
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [LookupEvents](#)。

Get-CTTrail

以下程式碼範例顯示如何使用 Get-CTTrail。

PowerShell V4 的工具

範例 1：傳回與您的帳戶目前區域相關聯的所有線索設定。

```
Get-CTTrail
```

範例 2：傳回指定線索的設定。

```
Get-CTTrail -TrailNameList trail1, trail2
```

範例 3：傳回在目前 shell 預設（在本例中為法蘭克福 (eu-central-1) 區域）以外區域中建立之指定線索的設定。

```
Get-CTTrail -TrailNameList trailABC, trailDEF -Region eu-central-1
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeTrails](#)。

Get-CTTrailStatus

以下程式碼範例顯示如何使用 Get-CTTrailStatus。

PowerShell V4 的工具

範例 1：傳回名為 'myExampleTrail' 之線索的狀態資訊。傳回的資料包含交付錯誤、Amazon SNS 和 Amazon S3 錯誤，以及開始和停止追蹤記錄時間的相關資訊。此範例假設線索是在與目前 shell 預設相同的區域中建立的。

```
Get-CTTrailStatus -Name myExampleTrail
```

範例 2：傳回在目前 shell 預設（在此情況下為法蘭克福 (eu-central-1) 區域）以外區域中建立之線索的狀態資訊。

```
Get-CTTrailStatus -Name myExampleTrail -Region eu-central-1
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetTrailStatus](#)。

New-CTTrail

以下程式碼範例顯示如何使用 New-CTTrail。

PowerShell V4 的工具

範例 1：建立將儲存貯體 'amzn-s3-demo-bucket' 用於日誌檔案儲存的線索。

```
New-CTTrail -Name "awscloudtrail-example" -S3BucketName "amzn-s3-demo-bucket"
```

範例 2：建立將使用儲存貯體 'amzn-s3-demo-bucket' 進行日誌檔案儲存的線索。代表日誌的 S3 物件會有 'mylogs' 的常見金鑰字首。當新日誌交付至儲存貯體時，通知將傳送至 SNS 主題 'mlog-deliverytopic'。此範例使用潑濺將參數值提供給 cmdlet。

```
$params = @{  
    Name="awscloudtrail-example"  
    S3BucketName="amzn-s3-demo-bucket"  
    S3KeyPrefix="mylogs"  
    SnsTopicName="mlog-deliverytopic"
```

```
}  
New-CTTrail @params
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateTrail](#)。

Remove-CTTrail

以下程式碼範例顯示如何使用 Remove-CTTrail。

PowerShell V4 的工具

範例 1：刪除指定的線索。在執行命令之前，系統會提示您進行確認。若要隱藏確認，請新增 -Force 切換參數。

```
Remove-CTTrail -Name "awscloudtrail-example"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteTrail](#)。

Start-CTLogging

以下程式碼範例顯示如何使用 Start-CTLogging。

PowerShell V4 的工具

範例 1：開始記錄名為 'myExampleTrail' 之線索的 AWS API 呼叫和日誌檔案交付。此範例假設線索是在與目前 shell 預設相同的區域中建立的。

```
Start-CTLogging -Name myExampleTrail
```

範例 2：開始記錄在目前 shell 預設（在此情況下為法蘭克福 (eu-central-1) 區域）以外區域中建立之線索的 AWS API 呼叫和日誌檔案交付。

```
Start-CTLogging -Name myExampleTrail -Region eu-central-1
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [StartLogging](#)。

Stop-CTLogging

以下程式碼範例顯示如何使用 Stop-CTLogging。

PowerShell V4 的工具

範例 1：暫停記錄名為 'myExampleTrail' 之線索的 AWS API 呼叫和日誌檔案交付。此範例假設線索是在與目前 shell 預設相同的區域中建立的。

```
Stop-CTLogging -Name myExampleTrail
```

範例 2：暫停記錄在目前 shell 預設（在此情況下為法蘭克福 (eu-central-1) 區域）以外區域中建立之線索的 AWS API 呼叫和日誌檔案交付。

```
Stop-CTLogging -Name myExampleTrail -Region eu-central-1
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [StopLogging](#)。

Update-CTTrail

以下程式碼範例顯示如何使用 Update-CTTrail。

PowerShell V4 的工具

範例 1：更新指定的線索，以便記錄全域服務事件（例如來自 IAM 的事件），並將後續日誌檔案的常見金鑰字首變更為「全域日誌」。

```
Update-CTTrail -Name "awscloudtrail-example" -IncludeGlobalServiceEvents $true -  
S3KeyPrefix "globallogs"
```

範例 2：更新指定的線索，以便將新日誌交付的通知傳送至指定的 SNS 主題。

```
Update-CTTrail -Name "awscloudtrail-example" -SnsTopicName "mlog-deliverytopic2"
```

範例 3：更新指定的線索，以便將日誌傳送到不同的儲存貯體。

```
Update-CTTrail -Name "awscloudtrail-example" -S3BucketName "otherlogs"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UpdateTrail](#)。

使用 Tools for PowerShell V4 的 CloudWatch 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 CloudWatch 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Get-CWAlarm

以下程式碼範例顯示如何使用 Get-CWAlarm。

PowerShell V4 的工具

範例 1：從 CloudWatch 傳回所有警示，包括複合和指標警示。

```
Get-CWAlarm -MaxRecords 1
```

輸出：

CompositeAlarms	MetricAlarms	NextToken
-----	-----	-----
	{MetricAlarms-01}	NextToken-01
	{MetricAlarms-02}	NextToken-02
	{MetricAlarms-03}	NextToken-03

範例 2：將 -AlarmType 參數設定為 CompositeAlarms 之後，僅傳回 CloudWatch 的複合警示資料。

```
Get-CWAlarm -AlarmType 'CompositeAlarms'
```

輸出：

CompositeAlarms	MetricAlarms	NextToken
{CompositeAlarms-01}		
{CompositeAlarms-02}		
{CompositeAlarms-03}		

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeAlarms](#)。

Get-CWDashboard

以下程式碼範例顯示如何使用 Get-CWDashboard。

PowerShell V4 的工具

範例 1：傳回指定儀表板內文的 arn。

```
Get-CWDashboard -DashboardName Dashboard1
```

輸出：

DashboardArn	DashboardBody
arn:aws:cloudwatch::123456789012:dashboard/Dashboard1	{...}

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetDashboard](#)。

Get-CWDashboardList

以下程式碼範例顯示如何使用 Get-CWDashboardList。

PowerShell V4 的工具

範例 1：傳回您帳戶的儀表板集合。

```
Get-CWDashboardList
```

輸出：

```
DashboardArn DashboardName LastModified      Size
-----
arn:...      Dashboard1      7/6/2017 8:14:15 PM 252
```

範例 2：傳回名稱開頭為 'dev' 之帳戶的儀表板集合。

```
Get-CWDashboardList -DashboardNamePrefix dev
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListDashboards](#)。

Remove-CWDashboard

以下程式碼範例顯示如何使用 Remove-CWDashboard。

PowerShell V4 的工具

範例 1：刪除指定的儀表板，在繼續之前提升確認。若要略過確認，請將 -Force 切換新增至命令。

```
Remove-CWDashboard -DashboardName Dashboard1
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteDashboards](#)。

Write-CWDashboard

以下程式碼範例顯示如何使用 Write-CWDashboard。

PowerShell V4 的工具

範例 1：建立或更新名為 'Dashboard1' 的儀表板，以並列包含兩個指標小工具。

```
$dashBody = @"
{
  "widgets":[
    {
```

```

        "type": "metric",
        "x": 0,
        "y": 0,
        "width": 12,
        "height": 6,
        "properties": {
            "metrics": [
                [
                    "AWS/EC2",
                    "CPUUtilization",
                    "InstanceId",
                    "i-012345"
                ]
            ],
            "period": 300,
            "stat": "Average",
            "region": "us-east-1",
            "title": "EC2 Instance CPU"
        }
    },
    {
        "type": "metric",
        "x": 12,
        "y": 0,
        "width": 12,
        "height": 6,
        "properties": {
            "metrics": [
                [
                    "AWS/S3",
                    "BucketSizeBytes",
                    "BucketName",
                    "amzn-s3-demo-bucket"
                ]
            ],
            "period": 86400,
            "stat": "Maximum",
            "region": "us-east-1",
            "title": "amzn-s3-demo-bucket bytes"
        }
    }
]
}
"@

```

```
Write-CWDashboard -DashboardName Dashboard1 -DashboardBody $dashBody
```

範例 2：建立或更新儀表板，將描述儀表板的內容導入 cmdlet。

```
$dashBody = @"
{
...
}
"@

$dashBody | Write-CWDashboard -DashboardName Dashboard1
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [PutDashboard](#)。

Write-CWMetricData

以下程式碼範例顯示如何使用 Write-CWMetricData。

PowerShell V4 的工具

範例 1：建立新的 MetricDatum 物件，並將其寫入 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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [PutMetricData](#)。

使用 Tools for PowerShell V4 的 CodeCommit 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 CodeCommit 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執程式碼的指示。

主題

- [動作](#)

動作

Get-CCBranch

以下程式碼範例顯示如何使用 Get-CCBranch。

PowerShell V4 的工具

範例 1：此範例會取得指定儲存庫之指定分支的相關資訊。

```
Get-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch
```

輸出：

BranchName	CommitId
-----	-----
MyNewBranch	7763222d...561fc9c9

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetBranch](#)。

Get-CCBranchList

以下程式碼範例顯示如何使用 Get-CCBranchList。

PowerShell V4 的工具

範例 1：此範例會取得指定儲存庫的分支名稱清單。

```
Get-CCBranchList -RepositoryName MyDemoRepo
```

輸出：

```
master
MyNewBranch
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListBranches](#)。

Get-CCRepository

以下程式碼範例顯示如何使用 Get-CCRepository。

PowerShell V4 的工具

範例 1：此範例會取得指定儲存庫的資訊。

```
Get-CCRepository -RepositoryName MyDemoRepo
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetRepository](#)。

Get-CCRepositoryBatch

以下程式碼範例顯示如何使用 Get-CCRepositoryBatch。

PowerShell V4 的工具

範例 1：此範例會確認找到和找不到哪些指定的儲存庫。


```
Get-CCRepositoryBatch -RepositoryName MyDemoRepo, MyNewRepo, AMissingRepo
```

輸出：

Repositories	RepositoriesNotFound
-----	-----
{MyDemoRepo, MyNewRepo}	{AMissingRepo}

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [BatchGetRepositories](#)。

Get-CCRepositoryList

以下程式碼範例顯示如何使用 Get-CCRepositoryList。

PowerShell V4 的工具

範例 1：此範例會依儲存庫名稱以遞增順序列出所有儲存庫。

```
Get-CCRepositoryList -Order Ascending -SortBy RepositoryName
```

輸出：

RepositoryId	RepositoryName
-----	-----
c7d0d2b0-ce40-4303-b4c3-38529EXAMPLE	MyDemoRepo
05f30c66-e3e3-4f91-a0cd-1c84aEXAMPLE	MyNewRepo

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListRepositories](#)。

New-CCBranch

以下程式碼範例顯示如何使用 New-CCBranch。

PowerShell V4 的工具

範例 1：此範例會建立具有指定儲存庫指定名稱和指定遞交 ID 的新分支。

```
New-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch -CommitId
7763222d...561fc9c9
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateBranch](#)。

New-CCRepository

以下程式碼範例顯示如何使用 New-CCRepository。

PowerShell V4 的工具

範例 1：此範例會建立具有指定名稱和指定描述的新儲存庫。

```
New-CCRepository -RepositoryName MyDemoRepo -RepositoryDescription "This is a
repository for demonstration purposes."
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateRepository](#)。

Remove-CCRepository

以下程式碼範例顯示如何使用 Remove-CCRepository。

PowerShell V4 的工具

範例 1：此範例強制刪除指定的儲存庫。命令會先提示確認再繼續。新增 `-Force` 參數以刪除儲存庫，而無需提示。

```
Remove-CCRepository -RepositoryName MyDemoRepo
```

輸出：

```
43ef2443-3372-4b12-9e78-65c27EXAMPLE
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteRepository](#)。

Update-CCDefaultBranch

以下程式碼範例顯示如何使用 `Update-CCDefaultBranch`。

PowerShell V4 的工具

範例 1：此範例會將指定儲存庫的預設分支變更為指定的分支。

```
Update-CCDefaultBranch -RepositoryName MyDemoRepo -DefaultBranchName MyNewBranch
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UpdateDefaultBranch](#)。

Update-CCRepositoryDescription

以下程式碼範例顯示如何使用 `Update-CCRepositoryDescription`。

PowerShell V4 的工具

範例 1：此範例會變更指定儲存庫的描述。

```
Update-CCRepositoryDescription -RepositoryName MyDemoRepo -RepositoryDescription  
"This is an updated description."
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [UpdateRepositoryDescription](#)。

Update-CCRepositoryName

以下程式碼範例顯示如何使用 Update-CCRepositoryName。

PowerShell V4 的工具

範例 1：此範例會變更指定儲存庫的名稱。

```
Update-CCRepositoryName -NewName MyDemoRepo2 -OldName MyDemoRepo
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [UpdateRepositoryName](#)。

使用 Tools for PowerShell V4 的 CodeDeploy 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 CodeDeploy 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Add-CDOnPremiseInstanceTag

以下程式碼範例顯示如何使用 Add-CDOnPremiseInstanceTag。

PowerShell V4 的工具

範例 1：此範例會為指定的現場部署執行個體新增具有指定金鑰和值的現場部署執行個體標籤。

```
Add-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" = "Name";  
"Value" = "CodeDeployDemo-OnPrem"}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [AddTagsToOnPremisesInstances](#)。

Get-CDApplication

以下程式碼範例顯示如何使用 Get-CDApplication。

PowerShell V4 的工具

範例 1：此範例會取得指定應用程式的相關資訊。

```
Get-CDApplication -ApplicationName CodeDeployDemoApplication
```

輸出：

ApplicationId LinkedToGitHub	ApplicationName	CreateTime
-----	-----	-----
e07fb938-091e-4f2f-8963-4d3e8EXAMPLE 9:49:48 PM False	CodeDeployDemoApplication	7/20/2015

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [GetApplication](#)。

Get-CDApplicationBatch

以下程式碼範例顯示如何使用 Get-CDApplicationBatch。

PowerShell V4 的工具

範例 1：此範例會取得指定應用程式的相關資訊。

```
Get-CDApplicationBatch -ApplicationName CodeDeployDemoApplication,  
CodePipelineDemoApplication
```

輸出：

ApplicationId LinkedToGitHub	ApplicationName	CreateTime

```

-----
-----
e07fb938-091e-4f2f-8963-4d3e8EXAMPLE    CodeDeployDemoApplication    7/20/2015
9:49:48 PM    False
1ecfd602-62f1-4038-8f0d-06688EXAMPLE    CodePipelineDemoApplication    8/13/2015
5:53:26 PM    False

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [BatchGetApplications](#)。

Get-CDApplicationList

以下程式碼範例顯示如何使用 Get-CDApplicationList。

PowerShell V4 的工具

範例 1：此範例會取得可用應用程式的清單。

```
Get-CDApplicationList
```

輸出：

```

CodeDeployDemoApplication
CodePipelineDemoApplication

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListApplications](#)。

Get-CDApplicationRevision

以下程式碼範例顯示如何使用 Get-CDApplicationRevision。

PowerShell V4 的工具

範例 1：此範例會取得指定應用程式修訂版的相關資訊。

```

$revision = Get-CDApplicationRevision -ApplicationName CodeDeployDemoApplication
-S3Location_Bucket amzn-s3-demo-bucket -Revision_RevisionType S3 -
S3Location_Key 5xd27EX.zip -S3Location_BundleType zip -S3Location_ETag
4565c1ac97187f190c1a90265EXAMPLE

```

```
Write-Output ("Description = " + $revision.RevisionInfo.Description + ",
RegisterTime = " + $revision.RevisionInfo.RegisterTime)
```

輸出：

```
Description = Application revision registered by Deployment ID: d-CX9CHN3EX,
RegisterTime = 07/20/2015 23:46:42
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetApplicationRevision](#)。

Get-CDApplicationRevisionList

以下程式碼範例顯示如何使用 Get-CDApplicationRevisionList。

PowerShell V4 的工具

範例 1：此範例會取得指定應用程式可用修訂的相關資訊。

```
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)
>> }
>> }
>> }
```

輸出：

```
Type = S3, Bucket = amzn-s3-demo-bucket, BundleType = zip, ETag =
4565c1ac97187f190c1a90265EXAMPLE, Key = 5xd27EX.zip
Type = GitHub, CommitId = f48933c3...76405362, Repository = MyGitHubUser/
CodeDeployDemoRepo
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListApplicationRevisions](#)。

Get-CDDeployment

以下程式碼範例顯示如何使用 Get-CDDeployment。

PowerShell V4 的工具

範例 1：此範例會取得指定部署的摘要資訊。

```
Get-CDDeployment -DeploymentId d-QZMRGSTEX
```

輸出：

```
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
```

範例 2：此範例會取得參與指定部署之執行個體狀態的相關資訊。

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).DeploymentOverview
```

輸出：

```
Failed      : 0
InProgress  : 0
Pending     : 0
Skipped     : 0
```



```
Succeeded : 3
```

範例 3：此範例會取得指定部署之應用程式修訂版的相關資訊。

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).Revision.S3Location
```

輸出：

```
Bucket      : amzn-s3-demo-bucket
BundleType  : zip
ETag        : cfbb81b304ee5e27efc21adaed3EXAMPLE
Key         : clzfqEX
Version     :
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetDeployment](#)。

Get-CDDeploymentBatch

以下程式碼範例顯示如何使用 Get-CDDeploymentBatch。

PowerShell V4 的工具

範例 1：此範例會取得指定部署的相關資訊。

```
Get-CDDeploymentBatch -DeploymentId d-QZMRGSTEX, d-RR0T5KTEX
```

輸出：

```
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

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [BatchGetDeployments](#)。

Get-CDDeploymentConfig

以下程式碼範例顯示如何使用 Get-CDDeploymentConfig。

PowerShell V4 的工具

範例 1：此範例會取得指定部署組態的摘要資訊。

```
Get-CDDeploymentConfig -DeploymentConfigName ThreeQuartersHealthy
```

輸出：

CreateTime	DeploymentConfigId	DeploymentConfigName
MinimumHealthyHosts		
-----	-----	-----

10/3/2014 4:32:30 PM	518a3950-d034-46a1-9d2c-3c949EXAMPLE	ThreeQuartersHealthy
Amazon.CodeDeploy.Model.MinimumHealthyHosts		

範例 2：此範例會取得指定部署組態定義的相關資訊。

```
Write-Output ((Get-CDDeploymentConfig -DeploymentConfigName
ThreeQuartersHealthy).MinimumHealthyHosts)
```

輸出：

Type	Value
----	-----
FLEET_PERCENT	75

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetDeploymentConfig](#)。

Get-CDDeploymentConfigList

以下程式碼範例顯示如何使用 Get-CDDeploymentConfigList。

PowerShell V4 的工具

範例 1：此範例會取得可用部署組態的清單。

```
Get-CDDeploymentConfigList
```

輸出：

```
ThreeQuartersHealthy
CodeDeployDefault.OneAtATime
CodeDeployDefault.AllAtOnce
CodeDeployDefault.HalfAtATime
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListDeploymentConfigs](#)。

Get-CDDeploymentGroup

以下程式碼範例顯示如何使用 Get-CDDeploymentGroup。

PowerShell V4 的工具

範例 1：此範例會取得指定部署群組的相關資訊。

```
Get-CDDeploymentGroup -ApplicationName CodeDeployDemoApplication -
DeploymentGroupName CodeDeployDemoFleet
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetDeploymentGroup](#)。

Get-CDDeploymentGroupList

以下程式碼範例顯示如何使用 Get-CDDeploymentGroupList。

PowerShell V4 的工具

範例 1：此範例會取得指定應用程式的部署群組清單。

```
Get-CDDeploymentGroupList -ApplicationName CodeDeployDemoApplication
```

輸出：

```
ApplicationName           DeploymentGroups
NextToken
-----
-----
CodeDeployDemoApplication {CodeDeployDemoFleet, CodeDeployProductionFleet}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListDeploymentGroups](#)。

Get-CDDeploymentInstance

以下程式碼範例顯示如何使用 Get-CDDeploymentInstance。

PowerShell V4 的工具

範例 1：此範例會取得指定部署之指定執行個體的相關資訊。

```
Get-CDDeploymentInstance -DeploymentId d-QZMRGSTEX -InstanceId i-254e22EX
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetDeploymentInstance](#)。

Get-CDDeploymentInstanceList

以下程式碼範例顯示如何使用 Get-CDDeploymentInstanceList。

PowerShell V4 的工具

範例 1：此範例會取得指定部署IDs 清單。

```
Get-CDDeploymentInstanceList -DeploymentId d-QZMRGSTEX
```

輸出：

```
i-254e22EX
i-274e22EX
i-3b4e22EX
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListDeploymentInstances](#)。

Get-CDDeploymentList

以下程式碼範例顯示如何使用 Get-CDDeploymentList。

PowerShell V4 的工具

範例 1：此範例會取得指定應用程式和部署群組的部署 IDs 清單。

```
Get-CDDeploymentList -ApplicationName CodeDeployDemoApplication -DeploymentGroupName  
CodeDeployDemoFleet
```

輸出：

```
d-QZMRGSTEX  
d-RR0T5KTEX
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListDeployments](#)。

Get-CDOnPremiseInstance

以下程式碼範例顯示如何使用 Get-CDOnPremiseInstance。

PowerShell V4 的工具

範例 1：此範例會取得指定內部部署執行個體的相關資訊。

```
Get-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

輸出：

```
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}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetOnPremisesInstance](#)。

Get-CDOnPremiseInstanceBatch

以下程式碼範例顯示如何使用 Get-CDOnPremiseInstanceBatch。

PowerShell V4 的工具

範例 1：此範例會取得指定內部部署執行個體的相關資訊。

```
Get-CDOnPremiseInstanceBatch -InstanceName AssetTag12010298EX, AssetTag12010298EX-2
```

輸出：

```
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}
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [BatchGetOnPremisesInstances](#)。

Get-CDOnPremiseInstanceList

以下程式碼範例顯示如何使用 Get-CDOnPremiseInstanceList。

PowerShell V4 的工具

範例 1：此範例會取得可用的現場部署執行個體名稱清單。

```
Get-CDOnPremiseInstanceList
```

輸出：

```
AssetTag12010298EX  
AssetTag12010298EX-2
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListOnPremisesInstances](#)。

New-CDApplication

以下程式碼範例顯示如何使用 New-CDApplication。

PowerShell V4 的工具

範例 1：此範例會建立具有指定名稱的新應用程式。

```
New-CDApplication -ApplicationName MyNewApplication
```

輸出：

```
f19e4b61-2231-4328-b0fd-e57f5EXAMPLE
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateApplication](#)。

New-CDDeployment

以下程式碼範例顯示如何使用 New-CDDeployment。

PowerShell V4 的工具

範例 1：此範例會使用指定的部署組態和應用程式修訂版，為指定的應用程式和部署群組建立新的部署。

```
New-CDDeployment -ApplicationName MyNewApplication -S3Location_Bucket amzn-s3-demo-  
bucket -S3Location_BundleType zip -DeploymentConfigName CodeDeployDefault.OneAtATime  
-DeploymentGroupName MyNewDeploymentGroup -IgnoreApplicationStopFailures $True -  
S3Location_Key aws-codedeploy_linux-master.zip -RevisionType S3
```

輸出：


```
d-ZHR0G7UEX
```

範例 2：此範例示範如何指定執行個體必須識別的 EC2 執行個體標籤群組，以便將執行個體包含在藍/綠部署的替代環境中。

```
New-CDDeployment -ApplicationName MyNewApplication -S3Location_Bucket amzn-s3-demo-  
bucket -S3Location_BundleType zip -DeploymentConfigName CodeDeployDefault.OneAtATime  
-DeploymentGroupName MyNewDeploymentGroup -IgnoreApplicationStopFailures $True  
-S3Location_Key aws-codedeploy_linux-master.zip -RevisionType S3 -Ec2TagSetList  
@(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{Key=
```

輸出：

```
d-ZHR0G7UEX
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateDeployment](#)。

New-CDDeploymentConfig

以下程式碼範例顯示如何使用 New-CDDeploymentConfig。

PowerShell V4 的工具

範例 1：此範例會使用指定的名稱和行為建立新的部署組態。

```
New-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts -  
MinimumHealthyHosts_Type HOST_COUNT -MinimumHealthyHosts_Value 2
```

輸出：

```
0f3e8187-44ef-42da-aeed-b6823EXAMPLE
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateDeploymentConfig](#)。

New-CDDeploymentGroup

以下程式碼範例顯示如何使用 New-CDDeploymentGroup。

PowerShell V4 的工具

範例 1：此範例會為指定的應用程式建立具有指定名稱的部署群組、Auto Scaling 群組、部署組態、標籤和服務角色。

```
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
```

輸出：

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

範例 2：此範例示範如何指定執行個體必須識別的 EC2 執行個體標籤群組，以便將執行個體包含在藍/綠部署的替代環境中。

```
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=
```

輸出：

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateDeploymentGroup](#)。

Register-CDApplicationRevision

以下程式碼範例顯示如何使用 Register-CDApplicationRevision。

PowerShell V4 的工具

範例 1：此範例會為指定的應用程式向指定的 Amazon S3 位置註冊應用程式修訂版。

```
Register-CDApplicationRevision -ApplicationName MyNewApplication -S3Location_Bucket  
amzn-s3-demo-bucket -S3Location_BundleType zip -S3Location_Key aws-  
codedeploy_linux-master.zip -Revision_RevisionType S3
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [RegisterApplicationRevision](#)。

Register-CDOnPremiseInstance

以下程式碼範例顯示如何使用 Register-CDOnPremiseInstance。

PowerShell V4 的工具

範例 1：此範例會使用指定的名稱和 IAM 使用者註冊現場部署執行個體。

```
Register-CDOnPremiseInstance -IamUserArn arn:aws:iam::80398EXAMPLE:user/  
CodeDeployDemoUser -InstanceName AssetTag12010298EX
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [RegisterOnPremisesInstance](#)。

Remove-CDApplication

以下程式碼範例顯示如何使用 Remove-CDApplication。

PowerShell V4 的工具

範例 1：此範例會刪除具有指定名稱的應用程式。命令會先提示確認再繼續。新增 -Force 參數以刪除應用程式，而無需提示。

```
Remove-CDApplication -ApplicationName MyNewApplication
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteApplication](#)。

Remove-CDDeploymentConfig

以下程式碼範例顯示如何使用 Remove-CDDeploymentConfig。

PowerShell V4 的工具

範例 1：此範例會刪除具有指定名稱的部署組態。命令會先提示確認再繼續。新增 `-Force` 參數以刪除部署組態，而無需提示。

```
Remove-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteDeploymentConfig](#)。

Remove-CDDeploymentGroup

以下程式碼範例顯示如何使用 `Remove-CDDeploymentGroup`。

PowerShell V4 的工具

範例 1：此範例會刪除具有指定應用程式指定名稱的部署群組。命令會先提示確認再繼續。新增 `-Force` 參數以刪除部署群組，而無需提示。

```
Remove-CDDeploymentGroup -ApplicationName MyNewApplication -DeploymentGroupName  
MyNewDeploymentGroup
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteDeploymentGroup](#)。

Remove-CDOnPremiseInstanceTag

以下程式碼範例顯示如何使用 `Remove-CDOnPremiseInstanceTag`。

PowerShell V4 的工具

範例 1：此範例會刪除具有指定名稱之現場部署執行個體的指定標籤。命令會先提示確認再繼續。新增 `-Force` 參數以刪除標籤，而無需提示。

```
Remove-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" =  
"Name"; "Value" = "CodeDeployDemo-OnPrem"}
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [RemoveTagsFromOnPremisesInstances](#)。

Stop-CDDeployment

以下程式碼範例顯示如何使用 Stop-CDDeployment。

PowerShell V4 的工具

範例 1：此範例會嘗試使用指定的部署 ID 停止部署。

```
Stop-CDDeployment -DeploymentId d-LJQNREYEX
```

輸出：

Status	StatusMessage
-----	-----
Pending	Stopping Pending. Stopping to schedule commands in the deployment instances

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [StopDeployment](#)。

Unregister-CDOnPremiseInstance

以下程式碼範例顯示如何使用 Unregister-CDOnPremiseInstance。

PowerShell V4 的工具

範例 1：此範例會使用指定的名稱取消註冊現場部署執行個體。

```
Unregister-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeregisterOnPremisesInstance](#)。

Update-CDApplication

以下程式碼範例顯示如何使用 Update-CDApplication。

PowerShell V4 的工具

範例 1：此範例會變更指定應用程式的名稱。

```
Update-CDApplication -ApplicationName MyNewApplication -NewApplicationName  
MyNewApplication-2
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UpdateApplication](#)。

Update-CDDeploymentGroup

以下程式碼範例顯示如何使用 Update-CDDeploymentGroup。

PowerShell V4 的工具

範例 1：此範例會變更指定應用程式之指定部署群組的名稱。

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication -  
CurrentDeploymentGroupName MyNewDeploymentGroup -NewDeploymentGroupName  
MyNewDeploymentGroup-2
```

範例 2：此範例示範如何指定執行個體必須識別的 EC2 執行個體標籤群組，以便將執行個體包含在藍/綠部署的替代環境中。

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication -  
CurrentDeploymentGroupName MyNewDeploymentGroup -NewDeploymentGroupName  
MyNewDeploymentGroup-2 -Ec2TagSetList  
(@(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{Key="
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [UpdateDeploymentGroup](#)。

使用 Tools for PowerShell V4 的 CodePipeline 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 CodePipeline 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Confirm-CPJob

以下程式碼範例顯示如何使用 Confirm-CPJob。

PowerShell V4 的工具

範例 1：此範例會取得指定任務的狀態。

```
Confirm-CPJob -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE -Nonce 3
```

輸出：

```
Value
-----
InProgress
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [AcknowledgeJob](#)。

Disable-CPStageTransition

以下程式碼範例顯示如何使用 Disable-CPStageTransition。

PowerShell V4 的工具

範例 1：此範例會停用指定管道中指定階段的傳入轉換。

```
Disable-CPStageTransition -PipelineName CodePipelineDemo -Reason "Disabling temporarily." -StageName Beta -TransitionType Inbound
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DisableStageTransition](#)。

Enable-CPStageTransition

以下程式碼範例顯示如何使用 Enable-CPStageTransition。

PowerShell V4 的工具

範例 1：此範例會啟用指定管道中指定階段的傳入轉換。

```
Enable-CPStageTransition -PipelineName CodePipelineDemo -StageName Beta -  
TransitionType Inbound
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [EnableStageTransition](#)。

Get-CPActionType

以下程式碼範例顯示如何使用 Get-CPActionType。

PowerShell V4 的工具

範例 1：此範例會取得指定擁有者所有可用動作的相關資訊。

```
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)
}

```

輸出：

```

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}

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListActionTypes](#)。

Get-CPActionableJobList

以下程式碼範例顯示如何使用 Get-CPActionableJobList。

PowerShell V4 的工具

範例 1：此範例會取得指定動作類別、擁有者、提供者、版本和查詢參數之所有可操作任務的相關資訊。

```

Get-CPActionableJobList -ActionTypeId_Category Build -ActionTypeId_Owner Custom
    -ActionTypeId_Provider MyCustomProviderName -ActionTypeId_Version 1 -QueryParam
    @{"ProjectName" = "MyProjectName"}

```

輸出：

AccountId	Data	Id
-----	-----	--
80398EXAMPLE	Amazon.CodePipeline.Model.JobData	0de392f5-712d-4f41-ace3-
f57a0EXAMPLE	3	

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [PollForJobs](#)。

Get-CPJobDetail

以下程式碼範例顯示如何使用 Get-CPJobDetail。

PowerShell V4 的工具

範例 1：此範例會取得指定任務的一般資訊。

```
Get-CPJobDetail -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE
```

輸出：

AccountId	Data	Id
-----	----	--
80398EXAMPLE	Amazon.CodePipeline.Model.JobData	
	f570dc12-5ef3-44bc-945a-6e133EXAMPLE	

範例 2：此範例會取得指定任務的詳細資訊。

```
$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)
```

輸出：

```
For Job f570dc12-5ef3-44bc-945a-6e133EXAMPLE:
  AccountId = 80398EXAMPLE
  Configuration:
  ActionTypeId:
    Category = Build
    Owner = Custom
    Provider = MyCustomProviderName
    Version = 1
  ArtifactCredentials:
    AccessKeyId = ASIAIEI3...IXI6YREX
    SecretAccessKey = cqAFDhEi...RdQyfa2u
    SessionToken = AQoDYXdz...5u+lsAU=
  InputArtifacts:
    MyApp
  OutputArtifacts:
    MyAppBuild
  PipelineContext:
    Name = Build
    PipelineName = CodePipelineDemo
    Stage = Build
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GetJobDetails](#)。

Get-CPPipeline

以下程式碼範例顯示如何使用 Get-CPPipeline。

PowerShell V4 的工具

範例 1：此範例會取得指定管道的一般資訊。

```
Get-CPPipeline -Name CodePipelineDemo -Version 1
```

輸出：

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {Source, Build, Beta, TestStage}
Version       : 1
```

範例 2：此範例會取得指定管道的詳細資訊。

```
$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)
```

```
}  
}
```

輸出：

```
Name = CodePipelineDemo  
RoleArn = arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole  
Version = 3  
ArtifactStore:  
  Location = amzn-s3-demo-bucket  
  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

```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetPipeline](#)。

Get-CPPipelineList

以下程式碼範例顯示如何使用 Get-CPPipelineList。

PowerShell V4 的工具

範例 1：此範例會取得可用管道的清單。

```
Get-CPPipelineList
```

輸出：

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListPipelines](#)。

Get-CPPipelineState

以下程式碼範例顯示如何使用 Get-CPPipelineState。

PowerShell V4 的工具

範例 1：此範例會取得指定管道階段的一般資訊。

```
Get-CPPipelineState -Name CodePipelineDemo
```

輸出：

```
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
```

範例 2：此範例會取得指定管道狀態的詳細資訊。

```
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)
    }
}

```

輸出：

```

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 =

```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetPipelineState](#)。

New-CPCustomActionType

以下程式碼範例顯示如何使用 New-CPCustomActionType。

PowerShell V4 的工具

範例 1：此範例會使用指定的屬性建立新的自訂動作。

```
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
```

輸出：

```
ActionConfigurationProperties : {ProjectName}
Id                           : Amazon.CodePipeline.Model.ActionTypeId
InputArtifactDetails         : Amazon.CodePipeline.Model.ArtifactDetails
OutputArtifactDetails        : Amazon.CodePipeline.Model.ArtifactDetails
Settings                     : Amazon.CodePipeline.Model.ActionTypeSettings
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateCustomActionType](#)。

New-CPPipeline

以下程式碼範例顯示如何使用 New-CPPipeline。

PowerShell V4 的工具

範例 1：此範例會建立具有指定設定的新管道。

```
$pipeline = New-Object Amazon.CodePipeline.Model.PipelineDeclaration

$sourceStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration
$deployStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration

$sourceStageActionOutputArtifact = New-Object
Amazon.CodePipeline.Model.OutputArtifact
$sourceStageActionOutputArtifact.Name = "MyApp"
```

```

$sourceStageAction.ActionTypeId = @{"Category" = "Source"; "Owner" = "AWS";
  "Provider" = "S3"; "Version" = 1}
$sourceStageAction.Configuration.Add("S3Bucket", "amzn-s3-demo-bucket")
$sourceStageAction.Configuration.Add("S3ObjectKey", "my-object-key-name.zip")
$sourceStageAction.OutputArtifacts.Add($sourceStageActionOutputArtifact)
$sourceStageAction.Name = "Source"

$deployStageActionInputArtifact = New-Object Amazon.CodePipeline.Model.InputArtifact
$deployStageActionInputArtifact.Name = "MyApp"

$deployStageAction.ActionTypeId = @{"Category" = "Deploy"; "Owner" = "AWS";
  "Provider" = "CodeDeploy"; "Version" = 1}
$deployStageAction.Configuration.Add("ApplicationName",
  "CodePipelineDemoApplication")
$deployStageAction.Configuration.Add("DeploymentGroupName", "CodePipelineDemoFleet")
$deployStageAction.InputArtifacts.Add($deployStageActionInputArtifact)
$deployStageAction.Name = "CodePipelineDemoFleet"

$sourceStage = New-Object Amazon.CodePipeline.Model.StageDeclaration
$deployStage = New-Object Amazon.CodePipeline.Model.StageDeclaration

$sourceStage.Name = "Source"
$deployStage.Name = "Beta"

$sourceStage.Actions.Add($sourceStageAction)
$deployStage.Actions.Add($deployStageAction)

$pipeline.ArtifactStore = @{"Location" = "amzn-s3-demo-bucket"; "Type" = "S3"}
$pipeline.Name = "CodePipelineDemo"
$pipeline.RoleArn = "arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole"
$pipeline.Stages.Add($sourceStage)
$pipeline.Stages.Add($deployStage)
$pipeline.Version = 1

New-CPPipeline -Pipeline $pipeline

```

輸出：

```

ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name           : CodePipelineDemo
RoleArn        : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages         : {Source, Beta}
Version        : 1

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreatePipeline](#)。

Remove-CPCustomActionType

以下程式碼範例顯示如何使用 Remove-CPCustomActionType。

PowerShell V4 的工具

範例 1：此範例會刪除指定的自訂動作。命令會先提示確認再繼續。新增 -Force 參數以刪除自訂動作，而無需提示。

```
Remove-CPCustomActionType -Category Build -Provider MyBuildProviderName -Version 1
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteCustomActionType](#)。

Remove-CPPipeline

以下程式碼範例顯示如何使用 Remove-CPPipeline。

PowerShell V4 的工具

範例 1：此範例會刪除指定的管道。命令會先提示確認再繼續。新增 -Force 參數，以在沒有提示的情況下刪除管道。

```
Remove-CPPipeline -Name CodePipelineDemo
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeletePipeline](#)。

Start-CPPipelineExecution

以下程式碼範例顯示如何使用 Start-CPPipelineExecution。

PowerShell V4 的工具

範例 1：此範例會開始執行指定的管道。

```
Start-CPPipelineExecution -Name CodePipelineDemo
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [StartPipelineExecution](#)。

Update-CPPipeline

以下程式碼範例顯示如何使用 Update-CPPipeline。

PowerShell V4 的工具

範例 1：此範例會使用指定的設定更新指定的現有管道。

```
$pipeline = New-Object Amazon.CodePipeline.Model.PipelineDeclaration

$sourceStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration
$deployStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration

$sourceStageAction.OutputArtifact = New-Object
    Amazon.CodePipeline.Model.OutputArtifact
$sourceStageAction.OutputArtifact.Name = "MyApp"

$sourceStageAction.ActionTypeId = @{"Category" = "Source"; "Owner" = "AWS";
    "Provider" = "S3"; "Version" = 1}
$sourceStageAction.Configuration.Add("S3Bucket", "amzn-s3-demo-bucket")
$sourceStageAction.Configuration.Add("S3ObjectKey", "my-object-key-name.zip")
$sourceStageAction.OutputArtifacts.Add($sourceStageAction.OutputArtifact)
$sourceStageAction.Name = "Source"

$deployStageAction.InputArtifact = New-Object Amazon.CodePipeline.Model.InputArtifact
$deployStageAction.InputArtifact.Name = "MyApp"

$deployStageAction.ActionTypeId = @{"Category" = "Deploy"; "Owner" = "AWS";
    "Provider" = "CodeDeploy"; "Version" = 1}
$deployStageAction.Configuration.Add("ApplicationName",
    "CodePipelineDemoApplication")
$deployStageAction.Configuration.Add("DeploymentGroupName", "CodePipelineDemoFleet")
$deployStageAction.InputArtifacts.Add($deployStageAction.InputArtifact)
$deployStageAction.Name = "CodePipelineDemoFleet"

$sourceStage = New-Object Amazon.CodePipeline.Model.StageDeclaration
$deployStage = New-Object Amazon.CodePipeline.Model.StageDeclaration
```

```
$sourceStage.Name = "MyInputFiles"
$deployStage.Name = "MyTestDeployment"

$sourceStage.Actions.Add($sourceStageAction)
$deployStage.Actions.Add($deployStageAction)

$pipeline.ArtifactStore = @{"Location" = "amzn-s3-demo-bucket"; "Type" = "S3"}
$pipeline.Name = "CodePipelineDemo"
$pipeline.RoleArn = "arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole"
$pipeline.Stages.Add($sourceStage)
$pipeline.Stages.Add($deployStage)
$pipeline.Version = 1

Update-CPipeline -Pipeline $pipeline
```

輸出：

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name           : CodePipelineDemo
RoleArn        : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages         : {InputFiles, TestDeployment}
Version        : 2
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UpdatePipeline](#)。

使用 Tools for PowerShell V4 的 Amazon Cognito 身分範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 Amazon Cognito Identity 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執程式碼的指示。

主題

- [動作](#)

動作

Get-CGIIdentityPool

以下程式碼範例顯示如何使用 Get-CGIIdentityPool。

PowerShell V4 的工具

範例 1：依特定身分集區 ID 擷取其相關資訊。

```
Get-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
```

輸出：

```

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

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeIdentityPool](#)。

Get-CGIIdentityPoolList

以下程式碼範例顯示如何使用 Get-CGIIdentityPoolList。

PowerShell V4 的工具

範例 1：擷取現有身分集區的清單。

```
Get-CGIIdentityPoolList
```

輸出：

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListIdentityPools](#)。

Get-CGIIdentityPoolRole

以下程式碼範例顯示如何使用 Get-CGIIdentityPoolRole。

PowerShell V4 的工具

範例 1：取得特定身分集區角色的相關資訊。

```
Get-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
```

輸出：

```

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

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetIdentityPoolRoles](#)。

New-CGIIdentityPool

以下程式碼範例顯示如何使用 New-CGIIdentityPool。

PowerShell V4 的工具

範例 1：建立新的身分集區，允許未驗證的身分。

```
New-CGIIdentityPool -AllowUnauthenticatedIdentities $true -IdentityPoolName CommonTests13
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateIdentityPool](#)。

Remove-CGIIdentityPool

以下程式碼範例顯示如何使用 Remove-CGIIdentityPool。

PowerShell V4 的工具

範例 1：刪除特定身分集區。

```
Remove-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteIdentityPool](#)。

Set-CGIIdentityPoolRole

以下程式碼範例顯示如何使用 Set-CGIIdentityPoolRole。

PowerShell V4 的工具

範例 1：將特定身分集區設定為具有未驗證的 IAM 角色。

```
Set-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1 -Role @{ "unauthenticated" = "arn:aws:iam::123456789012:role/CommonTests1Role" }
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [SetIdentityPoolRoles](#)。

Update-CGIIdentityPool

以下程式碼範例顯示如何使用 Update-CGIIdentityPool。

PowerShell V4 的工具

範例 1：更新一些身分集區屬性，在此情況下為身分集區的名稱。

```
Update-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1 -IdentityPoolName NewPoolName
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UpdateIdentityPool](#)。

AWS Config 使用 Tools for PowerShell V4 的範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 來執行動作和實作常見案例 AWS Config。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Add-CFGResourceTag

以下程式碼範例顯示如何使用 Add-CFGResourceTag。

PowerShell V4 的工具

範例 1：此範例會將指定的標籤與資源 ARN 建立關聯，在此案例中為 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"}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [TagResource](#)。

Get-CFGAggregateComplianceByConfigRuleList

以下程式碼範例顯示如何使用 Get-CFGAggregateComplianceByConfigRuleList。

PowerShell V4 的工具

範例 1：此範例會從指定組態規則的 ConfigurationAggregator 'kaju' 篩選中擷取詳細資訊，並展開/傳回規則的 'Compliance'。

```
Get-CFGAggregateComplianceByConfigRuleList -ConfigurationAggregatorName kaju
-Filters_ConfigRuleName ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK | Select-Object -
ExpandProperty Compliance
```

輸出：

```
ComplianceContributorCount      ComplianceType
-----
Amazon.ConfigService.Model.ComplianceContributorCount NON_COMPLIANT
```

範例 2：此範例會從指定的 ConfigurationAggregator 擷取詳細資訊，針對彙總工具涵蓋的所有區域篩選指定帳戶的詳細資訊，並進一步撤銷所有規則的合規。

```
Get-CFGAggregateComplianceByConfigRuleList -ConfigurationAggregatorName
kaju -Filters_AccountId 123456789012 | Select-Object ConfigRuleName,
@{N="Compliance";E={$_.Compliance.ComplianceType}}
```

輸出：

ConfigRuleName	Compliance
-----	-----
ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK	NON_COMPLIANT
ec2-instance-no-public-ip	NON_COMPLIANT
desired-instance-type	NON_COMPLIANT

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeAggregateComplianceByConfigRules](#)。

Get-CFGAggregateComplianceDetailsByConfigRule

以下程式碼範例顯示如何使用 Get-CFGAggregateComplianceDetailsByConfigRule。

PowerShell V4 的工具

範例 1：此範例會傳回評估結果，針對指定帳戶、彙總工具、區域和組態規則處於「COMPLIANT」狀態的 AWS Config 規則「desired-instance-type」，選取具有 resource-id 和 resource-type 的輸出

```
Get-CFGAggregateComplianceDetailsByConfigRule -AccountId 123456789012 -
AwsRegion eu-west-1 -ComplianceType COMPLIANT -ConfigRuleName desired-
instance-type -ConfigurationAggregatorName raju | Select-Object -
ExpandProperty EvaluationResultIdentifier | Select-Object -ExpandProperty
EvaluationResultQualifier
```

輸出：

ConfigRuleName	ResourceId	ResourceType
-----	-----	-----
desired-instance-type	i-0f1bf2f34c5678d12	AWS::EC2::Instance
desired-instance-type	i-0fd12dd3456789123	AWS::EC2::Instance

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet 參考》(V4)》中的 [GetAggregateComplianceDetailsByConfigRule](#)。

Get-CFGAggregateConfigRuleComplianceSummary

以下程式碼範例顯示如何使用 Get-CFGAggregateConfigRuleComplianceSummary。

PowerShell V4 的工具

範例 1：此範例會傳回指定彙總工具的不合規規則數目。

```
(Get-CFGAggregateConfigRuleComplianceSummary -ConfigurationAggregatorName
  raju).AggregateComplianceCounts.ComplianceSummary.NonCompliantResourceCount
```

輸出：

```
CapExceeded CappedCount
-----
False      5
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetAggregateConfigRuleComplianceSummary](#)。

Get-CFGAggregateDiscoveredResourceCount

以下程式碼範例顯示如何使用 Get-CFGAggregateDiscoveredResourceCount。

PowerShell V4 的工具

範例 1：此範例會傳回針對區域 us-east-1 篩選之指定彙總工具的資源計數。

```
Get-CFGAggregateDiscoveredResourceCount -ConfigurationAggregatorName Master -
  Filters_Region us-east-1
```

輸出：

```
GroupByKey GroupedResourceCounts NextToken TotalDiscoveredResources
-----
          {}                      455
```

範例 2：此範例會傳回指定彙總工具之篩選區域依 RESOURCE_TYPE 分組的資源計數。

```
Get-CFGAggregateDiscoveredResourceCount -ConfigurationAggregatorName Master -
Filters_Region us-east-1 -GroupByKey RESOURCE_TYPE |
Select-Object -ExpandProperty GroupedResourceCounts
```

輸出：

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GetAggregateDiscoveredResourceCounts](#)。

Get-CFGAggregateDiscoveredResourceList

以下程式碼範例顯示如何使用 Get-CFGAggregateDiscoveredResourceList。

PowerShell V4 的工具

範例 1：此範例會傳回 'Ireland' 彙總工具中彙總之指定資源類型的資源識別符。如需資源類型的清單，請參閱 https://docs.aws.amazon.com/sdkfornet/v3/apidocs/index.html?page=ConfigService/TConfigServiceResourceType.html&tocid=Amazon_ConfigService_ResourceType。

```
Get-CFGAggregateDiscoveredResourceList -ConfigurationAggregatorName Ireland -
ResourceType ([Amazon.ConfigService.ResourceType]::AWSAutoScalingAutoScalingGroup)
```

輸出：

```
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
```

範例 2：此範例會傳回使用 region us-east-1 篩選之指定彙總工具的 **AwsEC2SecurityGroup** 名為 'default' 的資源類型。

```
Get-CFGAggregateDiscoveredResourceList -ConfigurationAggregatorName raju -
ResourceType ([Amazon.ConfigService.ResourceType]::AWSEC2SecurityGroup) -
Filters_Region us-east-1 -Filters_ResourceName default
```

輸出：

```
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

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListAggregateDiscoveredResources](#)。

Get-CFGAggregateResourceConfig

以下程式碼範例顯示如何使用 Get-CFGAggregateResourceConfig。

PowerShell V4 的工具

範例 1：此範例會傳回指定資源的組態項目，並展開組態。

```

(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

```

輸出：

```

{"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"}]

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetAggregateResourceconfig-service](#)。

Get-CFGAggregateResourceConfigBatch

以下程式碼範例顯示如何使用 Get-CFGAggregateResourceConfigBatch。

PowerShell V4 的工具

範例 1：此範例會為特定彙總工具中存在的資源（已識別）擷取目前的組態項目。

```
$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
```

輸出：

```
BaseConfigurationItems  UnprocessedResourceIdentifiers
-----
{}                      {arn:aws:ec2:eu-west-1:123456789012:instance/
i-012e3cb4df567e8aa}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [BatchGetAggregateResourceconfig-service](#)。

Get-CFGAggregationAuthorizationList

以下程式碼範例顯示如何使用 Get-CFGAggregationAuthorizationList。

PowerShell V4 的工具

範例 1：此範例會擷取授予彙總工具的授權。

```
Get-CFGAggregationAuthorizationList
```

輸出：

```

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

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeAggregationAuthorizations](#)。

Get-CFGComplianceByConfigRule

以下程式碼範例顯示如何使用 Get-CFGComplianceByConfigRule。

PowerShell V4 的工具

範例 1：此範例會擷取規則 ebs-optimized-instance 的合規詳細資訊，其中規則目前沒有評估結果，因此會傳回 INSUFFICIENT_DATA

```
(Get-CFGComplianceByConfigRule -ConfigRuleName ebs-optimized-instance).Compliance
```

輸出：

```

ComplianceContributorCount ComplianceType
-----
INSUFFICIENT_DATA

```

範例 2：此範例會傳回規則 ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK 的不合規資源數目。

```
(Get-CFGComplianceByConfigRule -ConfigRuleName ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK -
ComplianceType NON_COMPLIANT).Compliance.ComplianceContributorCount
```

輸出：

```

CapExceeded CappedCount
-----
False      2

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeComplianceByConfigRule](#)。

Get-CFGComplianceByResource

以下程式碼範例顯示如何使用 Get-CFGComplianceByResource。

PowerShell V4 的工具

範例 1：此範例會檢查 'COMPLIANT' 合規類型的 **AWS::SSM::ManagedInstanceInventory** 資源類型。

```
Get-CFGComplianceByResource -ComplianceType COMPLIANT -ResourceType
AWS::SSM::ManagedInstanceInventory
```

輸出：

Compliance	ResourceId	ResourceType
-----	-----	-----
Amazon.ConfigService.Model.Compliance	i-0123bcf4b567890e3	
AWS::SSM::ManagedInstanceInventory		
Amazon.ConfigService.Model.Compliance	i-0a1234f6f5d6b78f7	
AWS::SSM::ManagedInstanceInventory		

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeComplianceByResource](#)。

Get-CFGComplianceDetailsByConfigRule

以下程式碼範例顯示如何使用 Get-CFGComplianceDetailsByConfigRule。

PowerShell V4 的工具

範例 1：此範例會取得規則 access-keys-rotated 的評估結果，並傳回依 compliance-type 分組的輸出

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-keys-rotated | Group-
Object ComplianceType
```

輸出：

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...

範例 2：此範例會查詢 COMPLIANT 資源之 access-keys-rotated 規則的合規詳細資訊。

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-
keys-rotated -ComplianceType COMPLIANT | ForEach-Object
{$_ .EvaluationResultIdentifier.EvaluationResultQualifier}
```

輸出：

ConfigRuleName	ResourceId	ResourceType
access-keys-rotated	BCAB1CDJ2LITAPVEW3JAH	AWS::IAM::User
access-keys-rotated	BCAB1CDJ2LITL3EHREM4Q	AWS::IAM::User

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GetComplianceDetailsByConfigRule](#)。

Get-CFGComplianceDetailsByResource

以下程式碼範例顯示如何使用 Get-CFGComplianceDetailsByResource。

PowerShell V4 的工具

範例 1：指定資源的此範例疏散結果。

```
Get-CFGComplianceDetailsByResource -ResourceId ABCD5STJ4EFGHIVEW6JAH -ResourceType
'AWS::IAM::User'
```

輸出：

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              :
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetComplianceDetailsByResource](#)。

Get-CFGComplianceSummaryByConfigRule

以下程式碼範例顯示如何使用 Get-CFGComplianceSummaryByConfigRule。

PowerShell V4 的工具

範例 1：此範例會傳回不合規的 Config 規則數目。

```
Get-CFGComplianceSummaryByConfigRule -Select
ComplianceSummary.NonCompliantResourceCount
```

輸出：

```
CapExceeded CappedCount
-----
False      9
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GetComplianceSummaryByConfigRule](#)。

Get-CFGComplianceSummaryByResourceType

以下程式碼範例顯示如何使用 Get-CFGComplianceSummaryByResourceType。

PowerShell V4 的工具

範例 1：此範例會傳回合規或不合規的資源數量，並將輸出轉換為 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
  }
}

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GetComplianceSummaryByResourceType](#)。

Get-CFGConfigRule

以下程式碼範例顯示如何使用 Get-CFGConfigRule。

PowerShell V4 的工具

範例 1：此範例列出具有所選屬性的帳戶組態規則。

```
Get-CFGConfigRule | Select-Object ConfigRuleName, ConfigRuleId, ConfigRuleArn,
ConfigRuleState
```

輸出：

ConfigRuleName	ConfigRuleId	ConfigRuleArn
	ConfigRuleState	
-----	-----	-----
ALB_REDIRECTION_CHECK	config-rule-12iyn3	arn:aws:config-
service:eu-west-1:123456789012:config-rule/config-rule-12iyn3	ACTIVE	
access-keys-rotated	config-rule-aospfr	arn:aws:config-
service:eu-west-1:123456789012:config-rule/config-rule-aospfr	ACTIVE	
autoscaling-group-elb-healthcheck-required	config-rule-cn1f2x	arn:aws:config-
service:eu-west-1:123456789012:config-rule/config-rule-cn1f2x	ACTIVE	

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeConfigRules](#)。

Get-CFGConfigRuleEvaluationStatus

以下程式碼範例顯示如何使用 Get-CFGConfigRuleEvaluationStatus。

PowerShell V4 的工具

範例 1：此範例會傳回指定組態規則的狀態資訊。

```
Get-CFGConfigRuleEvaluationStatus -ConfigRuleName root-account-mfa-enabled, vpc-
flow-logs-enabled
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeConfigRuleEvaluationStatus](#)。

Get-CFGConfigurationAggregatorList

以下程式碼範例顯示如何使用 Get-CFGConfigurationAggregatorList。

PowerShell V4 的工具

範例 1：此範例會傳回區域/帳戶的所有彙總工具。

```
Get-CFGConfigurationAggregatorList
```

輸出：

```
AccountAggregationSources      :  
  {Amazon.ConfigService.Model.AccountAggregationSource}  
ConfigurationAggregatorArn     : arn:aws:config-service:eu-  
west-1:123456789012:config-aggregator/config-aggregator-xabcalme  
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeConfigurationAggregators](#)。

Get-CFGConfigurationAggregatorSourcesStatus

以下程式碼範例顯示如何使用 Get-CFGConfigurationAggregatorSourcesStatus。

PowerShell V4 的工具

範例 1：此範例顯示指定彙總工具中來源的請求欄位。

```
Get-CFGConfigurationAggregatorSourcesStatus -ConfigurationAggregatorName raju |  
select SourceType, LastUpdateStatus, LastUpdateTime, SourceId
```

輸出：

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeConfigurationAggregatorSourcesStatus](#)。

Get-CFGConfigurationRecorder

以下程式碼範例顯示如何使用 Get-CFGConfigurationRecorder。

PowerShell V4 的工具

範例 1：此範例會傳回組態記錄器的詳細資訊。

```
Get-CFGConfigurationRecorder | Format-List
```

輸出：

```
Name           : default
RecordingGroup  : Amazon.ConfigService.Model.RecordingGroup
RoleARN        : arn:aws:iam::123456789012:role/aws-service-role/
config.amazonaws.com/AWSServiceRoleForConfig
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeConfigurationRecorders](#)。

Get-CFGConfigurationRecorderStatus

以下程式碼範例顯示如何使用 Get-CFGConfigurationRecorderStatus。

PowerShell V4 的工具

範例 1：此範例會傳回組態記錄器的狀態。

```
Get-CFGConfigurationRecorderStatus
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeConfigurationRecorderStatus](#)。

Get-CFGConformancePack

以下程式碼範例顯示如何使用 Get-CFGConformancePack。

PowerShell V4 的工具

範例 1：此範例列出所有一致性套件。

```
Get-CFGConformancePack
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeConformancePacks](#)。

Get-CFGDeliveryChannel

以下程式碼範例顯示如何使用 Get-CFGDeliveryChannel。

PowerShell V4 的工具

範例 1：此範例會擷取區域的交付管道，並顯示詳細資訊。

```
Get-CFGDeliveryChannel -Region eu-west-1 | Select-Object Name, S3BucketName,
S3KeyPrefix,
@{N="DeliveryFrequency";E={$_.ConfigSnapshotDeliveryProperties.DeliveryFrequency}}
```

輸出：

Name	S3BucketName	S3KeyPrefix	DeliveryFrequency
----	-----	-----	-----
default	config-bucket-NA	my	TwentyFour_Hours

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeDeliveryChannels](#)。

Get-CFGResourceTag

以下程式碼範例顯示如何使用 Get-CFGResourceTag。

PowerShell V4 的工具

範例 1：此範例列出指定資源的相關標籤

```
Get-CFGResourceTag -ResourceArn $rules[0].ConfigRuleArn
```

輸出：

Key	Value
---	-----
Version	1.3

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListTagsForResource](#)。

Remove-CFGConformancePack

以下程式碼範例顯示如何使用 Remove-CFGConformancePack。

PowerShell V4 的工具

範例 1：此範例會移除指定的一致性套件，以及套件的所有規則、修補動作和評估結果。

```
Remove-CFGConformancePack -ConformancePackName dono
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteConformancePack](#)。

Write-CFGConformancePack

以下程式碼範例顯示如何使用 Write-CFGConformancePack。

PowerShell V4 的工具

範例 1：此範例會建立一致性套件，從指定的 yaml 檔案擷取範本。

```
Write-CFGConformancePack -ConformancePackName dono -DeliveryS3Bucket amzn-s3-demo-
bucket -TemplateBody (Get-Content C:\windows\temp\template.yaml -Raw)
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [PutConformancePack](#)。

Write-CFGDeliveryChannel

以下程式碼範例顯示如何使用 Write-CFGDeliveryChannel。

PowerShell V4 的工具

範例 1：此範例會變更現有交付管道的 deliveryFrequency 屬性。

```
Write-CFGDeliveryChannel -ConfigSnapshotDeliveryProperties_DeliveryFrequency  
TwentyFour_Hours -DeliveryChannelName default -DeliveryChannel_S3BucketName amzn-  
s3-demo-bucket -DeliveryChannel_S3KeyPrefix my
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [PutDeliveryChannel](#)。

使用 Tools for PowerShell V4 的 Device Farm 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 Device Farm 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執程式碼的指示。

主題

- [動作](#)

動作

New-DFUpload

以下程式碼範例顯示如何使用 New-DFUpload。

PowerShell V4 的工具

範例 1：此範例會為 Android 應用程式建立 AWS Device Farm 上傳。您可以從 New-DFProject 或 Get-DFProjectList 的輸出取得專案 ARN。在 New-DFUpload 輸出中使用簽章的 URL，將檔案上傳至 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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateUpload](#)。

AWS Directory Service 使用 Tools for PowerShell V4 的範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 來執行動作和實作常見案例 AWS Directory Service。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Add-DSIpRoute

以下程式碼範例顯示如何使用 Add-DSIpRoute。

PowerShell V4 的工具

範例 1：此命令會移除指派給指定 Directory-id 的資源標籤

```
Add-DSIpRoute -DirectoryId d-123456ijkl -IpRoute @{CidrIp ="203.0.113.5/32"} -
UpdateSecurityGroupForDirectoryController $true
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [AddIpRoutes](#)。

Add-DSResourceTag

以下程式碼範例顯示如何使用 Add-DSResourceTag。

PowerShell V4 的工具

範例 1：此命令會將資源標籤新增至指定的 Directory-id


```
Add-DSResourceTag -ResourceId d-123456ijkl -Tag @{Key="myTag"; Value="mytgValue"}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [AddTagsToResource](#)。

Approve-DSTrust

以下程式碼範例顯示如何使用 Approve-DSTrust。

PowerShell V4 的工具

範例 1：此範例會呼叫指定 Trustid 的 AWS Directory Service VerifyTrust API 操作。

```
Approve-DSTrust -TrustId t-9067157123
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [VerifyTrust](#)。

Confirm-DSSharedDirectory

以下程式碼範例顯示如何使用 Confirm-DSSharedDirectory。

PowerShell V4 的工具

範例 1：此範例接受從目錄擁有者傳送的目錄共用請求 AWS 帳戶。

```
Confirm-DSSharedDirectory -SharedDirectoryId d-9067012345
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [AcceptSharedDirectory](#)。

Connect-DSDirectory

以下程式碼範例顯示如何使用 Connect-DSDirectory。

PowerShell V4 的工具

範例 1：此範例會建立 AD Connector 以連線至內部部署目錄。

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ConnectDirectory](#)。

Deny-DSSharedDirectory

以下程式碼範例顯示如何使用 Deny-DSSharedDirectory。

PowerShell V4 的工具

範例 1：此範例會拒絕從目錄擁有者帳戶傳送的目錄共用請求。

```
Deny-DSSharedDirectory -SharedDirectoryId d-9067012345
```

輸出：

```
d-9067012345
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [RejectSharedDirectory](#)。

Disable-DSDirectoryShare

以下程式碼範例顯示如何使用 Disable-DSDirectoryShare。

PowerShell V4 的工具

範例 1：此範例會停止目錄擁有者和消費者帳戶之間的目錄共用。

```
Disable-DSDirectoryShare -DirectoryId d-123456ijkl -UnshareTarget_Id 123456784321 -  
UnshareTarget_Type ACCOUNT
```

輸出：

```
d-9067012345
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UnshareDirectory](#)。

Disable-DSLDPAS

以下程式碼範例顯示如何使用 Disable-DSLDPAS。

PowerShell V4 的工具

範例 1：此範例會停用指定目錄的 LDAP 安全呼叫。

```
Disable-DSLDPAS -DirectoryId d-123456ijkl -Type Client
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DisableLDAP](#)。

Disable-DSRadius

以下程式碼範例顯示如何使用 Disable-DSRadius。

PowerShell V4 的工具

範例 1：此範例會停用針對 AD Connector 或 Microsoft AD 目錄設定的 RADIUS 伺服器。

```
Disable-DSRadius -DirectoryId d-123456ijkl
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DisableRadius](#)。

Disable-DSSso

以下程式碼範例顯示如何使用 Disable-DSSso。

PowerShell V4 的工具

範例 1：此範例會停用目錄的單一登入。

```
Disable-DSSso -DirectoryId d-123456ijkl
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DisableSso](#)。

Enable-DSDirectoryShare

以下程式碼範例顯示如何使用 Enable-DSDirectoryShare。

PowerShell V4 的工具

範例 1：此範例使用交握方法與另一個 AWS 帳戶共用您 AWS 帳戶中指定的目錄。

```
Enable-DSDirectoryShare -DirectoryId d-123456ijkl -ShareTarget_Id 123456784321 -  
ShareMethod HANDSHAKE -ShareTarget_Type ACCOUNT
```

輸出：

```
d-9067012345
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ShareDirectory](#)。

Enable-DSLDAPs

以下程式碼範例顯示如何使用 Enable-DSLDAPs。

PowerShell V4 的工具

範例 1：此範例會啟用特定目錄的開關，以一律使用 LDAP 安全呼叫。

```
Enable-DSLDAPs -DirectoryId d-123456ijkl -Type Client
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [EnableLDAPs](#)。

Enable-DSRadius

以下程式碼範例顯示如何使用 Enable-DSRadius。

PowerShell V4 的工具

範例 1：此範例使用 AD Connector 或 Microsoft AD 目錄提供的 RADIUS 伺服器組態啟用多重要素驗證 (MFA)。

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [EnableRadius](#)。

Enable-DSSso

以下程式碼範例顯示如何使用 Enable-DSSso。

PowerShell V4 的工具

範例 1：此範例會啟用目錄的單一登入。

```
Enable-DSSso -DirectoryId d-123456ijkl
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [EnableSso](#)。

Get-DSCertificate

以下程式碼範例顯示如何使用 Get-DSCertificate。

PowerShell V4 的工具

範例 1：此範例顯示註冊安全 LDAP 連線之憑證的相關資訊。

```
Get-DSCertificate -DirectoryId d-123456ijkl -CertificateId c-906731e34f
```

輸出：

```
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.
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeCertificate](#)。

Get-DSCertificateList

以下程式碼範例顯示如何使用 Get-DSCertificateList。

PowerShell V4 的工具

範例 1：此範例列出為指定目錄的安全 LDAP 連線註冊的所有憑證。

```
Get-DSCertificateList -DirectoryId d-123456ijkl
```

輸出：

CertificateId	CommonName	ExpiryDateTime	State
-----	-----	-----	-----
c-906731e34f	contoso-EC2AMAZ-CTGG2NM-CA	4/15/2025 6:34:15 PM	Registered

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListCertificates](#)。

Get-DSConditionalForwarder

以下程式碼範例顯示如何使用 Get-DSConditionalForwarder。

PowerShell V4 的工具

範例 1：此命令會取得指定 Directory-id 的所有設定條件式轉寄站。

```
Get-DSConditionalForwarder -DirectoryId d-123456ijkl
```

輸出：

DnsIpAddrs	RemoteDomainName	ReplicationScope
-----	-----	-----
{172.31.77.239}	contoso.com	Domain

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeConditionalForwarders](#)。

Get-DSDirectory

以下程式碼範例顯示如何使用 Get-DSDirectory。

PowerShell V4 的工具

範例 1：此命令會取得屬於此帳戶之目錄的相關資訊。

```
Get-DSDirectory | Select-Object DirectoryId, Name, DnsIpAddrs, Type
```

輸出：

DirectoryId	Name	DnsIpAddrs	Type
-----	----	-----	----
d-123456abcd	abcd.example.com	{172.31.74.189, 172.31.13.145}	SimpleAD
d-123456efgh	wifi.example.com	{172.31.16.108, 172.31.10.56}	ADConnector
d-123456ijkl	lan2.example.com	{172.31.10.56, 172.31.16.108}	MicrosoftAD

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeDirectories](#)。

Get-DSDirectoryLimit

以下程式碼範例顯示如何使用 Get-DSDirectoryLimit。

PowerShell V4 的工具

範例 1：此範例會分派 us-east-1 區域的目錄限制資訊。

```
Get-DSDirectoryLimit -Region us-east-1
```

輸出：

```
CloudOnlyDirectoriesCurrentCount : 1
CloudOnlyDirectoriesLimit        : 10
CloudOnlyDirectoriesLimitReached : False
CloudOnlyMicrosoftADCurrentCount : 1
CloudOnlyMicrosoftADLimit       : 20
CloudOnlyMicrosoftADLimitReached : False
ConnectedDirectoriesCurrentCount : 1
ConnectedDirectoriesLimit        : 10
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetDirectoryLimits](#)。

Get-DSDomainControllerList

以下程式碼範例顯示如何使用 Get-DSDomainControllerList。

PowerShell V4 的工具

範例 1：此命令會取得針對提及的 directory-id 啟動的網域控制站詳細清單

```
Get-DSDomainControllerList -DirectoryId d-123456ijkl
```

輸出：

```
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

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeDomainControllers](#)。

Get-DSEventTopic

以下程式碼範例顯示如何使用 Get-DSEventTopic。

PowerShell V4 的工具

範例 1：此命令顯示目錄狀態變更時所設定 SNS 主題的通知資訊。

```
Get-DSEventTopic -DirectoryId d-123456ijkl
```

輸出：

```

CreatedDateTime : 12/13/2019 11:15:32 AM
DirectoryId     : d-123456ijkl
Status         : Registered
TopicArn       : arn:aws:sns:us-east-1:123456781234:snstopicname
TopicName      : snstopicname

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeEventTopics](#)。

Get-DSIpRouteList

以下程式碼範例顯示如何使用 Get-DSIpRouteList。

PowerShell V4 的工具

範例 1：此命令會取得在目錄 IP 路由中設定的公有 IP 地址區塊

```
Get-DSIpRouteList -DirectoryId d-123456ijkl
```

輸出：

```
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 :
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListIpRoutes](#)。

Get-DSLdapSetting

以下程式碼範例顯示如何使用 Get-DSLdapSetting。

PowerShell V4 的工具

範例 1：此範例說明指定目錄的 LDAP 安全性狀態。

```
Get-DSLdapSetting -DirectoryId d-123456ijkl
```

輸出：

```
LastUpdatedDateTime  LDAPStatus  LDAPStatusReason
-----
4/15/2020 6:51:03 PM Enabled      LDAPs is enabled successfully.
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeLDAPSettings](#)。

Get-DSLogSubscriptionList

以下程式碼範例顯示如何使用 Get-DSLogSubscriptionList。

PowerShell V4 的工具

範例 1：此命令會取得指定 directory-id 的日誌訂閱資訊

```
Get-DSLogSubscriptionList -DirectoryId d-123456ijkl
```

輸出：

```
DirectoryId  LogGroupName
SubscriptionCreatedDateTime
-----
d-123456ijkl /aws/directoryservice/d-123456ijkl-lan2.example.com 12/14/2019 9:05:23
AM
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListLogSubscriptions](#)。

Get-DSResourceTag

以下程式碼範例顯示如何使用 Get-DSResourceTag。

PowerShell V4 的工具

範例 1：此命令會取得指定目錄的所有標籤。

```
Get-DSResourceTag -ResourceId d-123456ijkl
```

輸出：

```
Key    Value
---    -
myTag  myTagValue
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListTagsForResource](#)。

Get-DSSchemaExtension

以下程式碼範例顯示如何使用 Get-DSSchemaExtension。

PowerShell V4 的工具

範例 1：此範例列出套用至 Microsoft AD 目錄的所有結構描述延伸。

```
Get-DSSchemaExtension -DirectoryId d-123456ijkl
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListSchemaExtensions](#)。

Get-DSSharedDirectory

以下程式碼範例顯示如何使用 Get-DSSharedDirectory。

PowerShell V4 的工具

範例 1：此範例會取得您 AWS 帳戶的共用目錄

```
Get-DSSharedDirectory -OwnerDirectoryId d-123456ijkl -SharedDirectoryId d-9067012345
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeSharedDirectories](#)。

Get-DSSnapshot

以下程式碼範例顯示如何使用 Get-DSSnapshot。

PowerShell V4 的工具

範例 1：此命令會取得屬於此帳戶之指定目錄快照的相關資訊。

```
Get-DSSnapshot -DirectoryId d-123456ijkl
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeSnapshots](#)。

Get-DSSnapshotLimit

以下程式碼範例顯示如何使用 Get-DSSnapshotLimit。

PowerShell V4 的工具

範例 1：此命令會取得指定目錄的手動快照限制。

```
Get-DSSnapshotLimit -DirectoryId d-123456ijkl
```

輸出：

```
ManualSnapshotsCurrentCount ManualSnapshotsLimit ManualSnapshotsLimitReached
-----
0                           5                           False
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetSnapshotLimits](#)。

Get-DSTrust

以下程式碼範例顯示如何使用 Get-DSTrust。

PowerShell V4 的工具

範例 1：此命令會取得為指定 directory-id 建立的信任關係資訊。

```
Get-DSTrust -DirectoryId d-123456abcd
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeTrusts](#)。

New-DSAlias

以下程式碼範例顯示如何使用 New-DSAlias。

PowerShell V4 的工具

範例 1：此命令會建立目錄的別名，並將別名指派給指定的 directory-id。

```
New-DSAlias -DirectoryId d-123456ijkl -Alias MyOrgName
```

輸出：

```
Alias      DirectoryId
-----
myorgname d-123456ijkl
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateAlias](#)。

New-DSComputer

以下程式碼範例顯示如何使用 New-DSComputer。

PowerShell V4 的工具

範例 1：此範例會建立新的 Active Directory 電腦物件。

```
New-DSComputer -DirectoryId d-123456ijkl -ComputerName ADMemberServer -Password $Password
```

輸出：

```
ComputerAttributes      ComputerId
ComputerName
-----
-----
{WindowsSamName, DistinguishedName} S-1-5-21-1191241402-978882507-2717148213-1662
ADMemberServer
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [CreateComputer](#)。

New-DSConditionalForwarder

以下程式碼範例顯示如何使用 New-DSConditionalForwarder。

PowerShell V4 的工具

範例 1：此範例會在指定的 AWS Directory-id 中建立條件式轉送器。

```
New-DSConditionalForwarder -DirectoryId d-123456ijkl -DnsIpAddress 172.31.36.96,172.31.10.56 -RemoteDomainName contoso.com
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [CreateConditionalForwarder](#)。

New-DSDirectory

以下程式碼範例顯示如何使用 New-DSDirectory。

PowerShell V4 的工具

範例 1：此範例會建立新的 Simple AD 目錄。

```
New-DSDirectory -Name corp.example.com -Password $Password -Size Small -VpcSettings_VpcId vpc-123459d -VpcSettings_SubnetIds subnet-1234kkaa,subnet-5678ffbb
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateDirectory](#)。

New-DSLogSubscription

以下程式碼範例顯示如何使用 New-DSLogSubscription。

PowerShell V4 的工具

範例 1：此範例會建立訂閱，將即時 Directory Service 網域控制站安全日誌轉送至您 中指定的 Amazon CloudWatch 日誌群組 AWS 帳戶。

```
New-DSLogSubscription -DirectoryId d-123456ijkl -LogGroupName /aws/directoryservice/d-123456ijkl-lan2.example.com
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateLogSubscription](#)。

New-DSMicrosoftAD

以下程式碼範例顯示如何使用 New-DSMicrosoftAD。

PowerShell V4 的工具

範例 1：此範例會在 中建立新的 Microsoft AD Directory AWS 雲端。

```
New-DSMicrosoftAD -Name corp.example.com -Password $Password -edition Standard -  
VpcSettings_VpcId vpc-123459d -VpcSettings_SubnetIds subnet-1234kkaa,subnet-5678ffbb
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateMicrosoftAD](#)。

New-DSSnapshot

以下程式碼範例顯示如何使用 New-DSSnapshot。

PowerShell V4 的工具

範例 1：此範例會建立目錄快照

```
New-DSSnapshot -DirectoryId d-123456ijkl
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateSnapshot](#)。

New-DSTrust

以下程式碼範例顯示如何使用 New-DSTrust。

PowerShell V4 的工具

範例 1：此範例會在 AWS Managed Microsoft AD 目錄與現有的內部部署 Microsoft Active Directory 之間建立雙向全森林信任。

```
New-DSTrust -DirectoryId d-123456ijkl -RemoteDomainName contoso.com -TrustDirection  
Two-Way -TrustType Forest -TrustPassword $Password -ConditionalForwarderIpAddr  
172.31.36.96
```

輸出：

```
t-9067157123
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateTrust](#)。

Register-DSCertificate

以下程式碼範例顯示如何使用 Register-DSCertificate。

PowerShell V4 的工具

範例 1：此範例會註冊安全 LDAP 連線的憑證。

```
$Certificate = Get-Content contoso.cer -Raw  
Register-DSCertificate -DirectoryId d-123456ijkl -CertificateData $Certificate
```

輸出：

```
c-906731e350
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [RegisterCertificate](#)。

Register-DSEventTopic

以下程式碼範例顯示如何使用 Register-DSEventTopic。

PowerShell V4 的工具

範例 1：此範例會將目錄作為發佈者與 SNS 主題建立關聯。

```
Register-DSEventTopic -DirectoryId d-123456ijkl -TopicName snstopicname
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [RegisterEventTopic](#)。

Remove-DSConditionalForwarder

以下程式碼範例顯示如何使用 Remove-DSConditionalForwarder。

PowerShell V4 的工具

範例 1：此範例會移除已為您的 AWS 分類設定的條件式轉送器。

```
Remove-DSConditionalForwarder -DirectoryId d-123456ijkl -RemoteDomainName  
contoso.com
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteConditionalForwarder](#)。

Remove-DSDirectory

以下程式碼範例顯示如何使用 Remove-DSDirectory。

PowerShell V4 的工具

範例 1：此範例會刪除 AWS Directory 服務目錄 (Simple AD/Microsoft AD/AD Connector)

```
Remove-DSDirectory -DirectoryId d-123456ijkl
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteDirectory](#)。

Remove-DSIpRoute

以下程式碼範例顯示如何使用 Remove-DSIpRoute。

PowerShell V4 的工具

範例 1：此命令會從 Directory-id 的設定 IP 路由中移除指定的 IP。

```
Remove-DSIpRoute -DirectoryId d-123456ijkl -CidrIp 203.0.113.5/32
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [RemoveIpRoutes](#)。

Remove-DSLogSubscription

以下程式碼範例顯示如何使用 Remove-DSLogSubscription。

PowerShell V4 的工具

範例 1：此命令會移除指定 Directory-id 的日誌訂閱

```
Remove-DSLogSubscription -DirectoryId d-123456ijkl
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteLogSubscription](#)。

Remove-DSResourceTag

以下程式碼範例顯示如何使用 Remove-DSResourceTag。

PowerShell V4 的工具

範例 1：此命令會移除指派給指定 Directory-id 的資源標籤

```
Remove-DSResourceTag -ResourceId d-123456ijkl -TagKey myTag
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [RemoveTagsFromResource](#)。

Remove-DSSnapshot

以下程式碼範例顯示如何使用 Remove-DSSnapshot。

PowerShell V4 的工具

範例 1：此範例會移除手動建立的快照。

```
Remove-DSSnapshot -SnapshotId s-9068b488kc
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteSnapshot](#)。

Remove-DSTrust

以下程式碼範例顯示如何使用 Remove-DSTrust。

PowerShell V4 的工具

範例 1：此範例會移除 AWS Managed AD Directory 與外部網域之間存在的信任關係。

```
Get-DSTrust -DirectoryId d-123456ijkl -Select Trusts.TrustId | Remove-DSTrust
```

輸出：

```
t-9067157123
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteTrust](#)。

Reset-DSUserPassword

以下程式碼範例顯示如何使用 Reset-DSUserPassword。

PowerShell V4 的工具

範例 1：此範例會在 AWS Managed microsoft AD 或 Simple AD Directory 中重設名為 ADUser 的 Active Directory 使用者密碼

```
Reset-DSUserPassword -UserName ADUser -DirectoryId d-123456ijkl -NewPassword $Password
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ResetUserPassword](#)。

Restore-DSFromSnapshot

以下程式碼範例顯示如何使用 Restore-DSFromSnapshot。

PowerShell V4 的工具

範例 1：此範例使用現有的目錄快照還原目錄。

```
Restore-DSFromSnapshot -SnapshotId s-9068b488kc
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [RestoreFromSnapshot](#)。

Set-DSDomainControllerCount

以下程式碼範例顯示如何使用 Set-DSDomainControllerCount。

PowerShell V4 的工具

範例 1：此範例會將指定 directory-id 的網域控制器數目設定為 3。

```
Set-DSDomainControllerCount -DirectoryId d-123456ijkl -DesiredNumber 3
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UpdateNumberOfDomainControllers](#)。

Start-DSSchemaExtension

以下程式碼範例顯示如何使用 Start-DSSchemaExtension。

PowerShell V4 的工具

範例 1：此範例會將結構描述延伸套用至 Microsoft AD 目錄。

```
$ldif = Get-Content D:\Users\Username\Downloads\ExtendedSchema.ldf -Raw
Start-DSSchemaExtension -DirectoryId d-123456ijkl -
CreateSnapshotBeforeSchemaExtension $true -Description ManagedADSchemaExtension -
LdifContent $ldif
```

輸出：

```
e-9067306643
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [StartSchemaExtension](#)。

Stop-DSSchemaExtension

以下程式碼範例顯示如何使用 Stop-DSSchemaExtension。

PowerShell V4 的工具

範例 1：此範例會取消 Microsoft AD 目錄的進行中結構描述延伸。

```
Stop-DSSchemaExtension -DirectoryId d-123456ijkl -SchemaExtensionId e-9067306643
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CancelSchemaExtension](#)。

Unregister-DSCertificate

以下程式碼範例顯示如何使用 Unregister-DSCertificate。

PowerShell V4 的工具

範例 1：此範例會從系統刪除已註冊安全 LDAP 連線的憑證。

```
Unregister-DSCertificate -DirectoryId d-123456ijkl -CertificateId c-906731e34f
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeregisterCertificate](#)。

Unregister-DSEventTopic

以下程式碼範例顯示如何使用 Unregister-DSEventTopic。

PowerShell V4 的工具

範例 1：此範例會將指定目錄移除為指定 SNS 主題的發佈者。

```
Unregister-DSEventTopic -DirectoryId d-123456ijkl -TopicName snstopicname
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeregisterEventTopic](#)。

Update-DSConditionalForwarder

以下程式碼範例顯示如何使用 Update-DSConditionalForwarder。

PowerShell V4 的工具

範例 1：此範例會更新已為您的 AWS 目錄設定的條件式轉送器。

```
Update-DSConditionalForwarder -DirectoryId d-123456ijkl -DnsIpAddress 172.31.36.96,172.31.16.108 -RemoteDomainName contoso.com
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [UpdateConditionalForwarder](#)。

Update-DSRadius

以下程式碼範例顯示如何使用 Update-DSRadius。

PowerShell V4 的工具

範例 1：此範例會更新 AD Connector 或 Microsoft AD 目錄的 RADIUS 伺服器資訊。

```
Update-DSRadius -DirectoryId d-123456ijkl -RadiusSettings_RadiusRetry 3
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UpdateRadius](#)。

Update-DSTrust

以下程式碼範例顯示如何使用 Update-DSTrust。

PowerShell V4 的工具

範例 1：此範例會將指定 trust-id 的 SelectiveAuth 參數從已停用更新為已啟用。

```
Update-DSTrust -TrustId t-9067157123 -SelectiveAuth Enabled
```

輸出：

RequestId	TrustId
-----	-----
138864a7-c9a8-4ad1-a828-eae479e85b45	t-9067157123

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UpdateTrust](#)。

AWS DMS 使用 Tools for PowerShell V4 的範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 來執行動作和實作常見案例 AWS DMS。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

New-DMSReplicationTask

以下程式碼範例顯示如何使用 New-DMSReplicationTask。

PowerShell V4 的工具

範例 1：此範例會建立新的 AWS Database Migration Service 複寫任務，使用 CdcStartTime 而非 CdcStartPosition。MigrationType 設定為 "full-load-and-cdc"，表示目標資料表必須為空白。新任務會加上具有 Stage 金鑰和 Test 金鑰值的標籤。如需此 cmdlet 使用之值的詳細資訊，請參閱《AWS Database Migration Service 使用者指南》中的建立任務 (https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Tasks.Creating.html)。

```
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"
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateReplicationTask](#)。

使用 Tools for PowerShell V4 的 DynamoDB 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 DynamoDB 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執程式碼的指示。

主題

- [動作](#)

動作

Add-DDBIndexSchema

以下程式碼範例顯示如何使用 Add-DDBIndexSchema。

PowerShell V4 的工具

範例 1：建立空的 TableSchema 物件，並在將 TableSchema 物件寫入管道之前新增新的本機次要索引定義。

```
$schema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
"LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
$schema = New-DDBTableSchema
```

輸出：

AttributeSchema	KeySchema
LocalSecondaryIndexSchema	
-----	-----
{LastPostDateTime}	
{LastPostIndex}	{}

範例 2：將新的本機次要索引定義新增至提供的 TableSchema 物件，然後再將 TableSchema 物件寫入管道。TableSchema 物件也可以使用 -Schema 參數提供。

```
New-DDBTableSchema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
"LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
```

輸出：

AttributeSchema	KeySchema
LocalSecondaryIndexSchema	
-----	-----

{LastPostDateTime}	{}
{LastPostIndex}	

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [Add-DDBIndexSchema](#)。

Add-DDBKeySchema

以下程式碼範例顯示如何使用 Add-DDBKeySchema。

PowerShell V4 的工具

範例 1：建立空的 TableSchema 物件，並使用指定的金鑰資料將金鑰和屬性定義項目新增至其中，然後再將 TableSchema 物件寫入管道。根據預設，金鑰類型宣告為 'HASH'；使用值為 'RANGE' 的 -KeyType parameter 宣告範圍金鑰。

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
```

輸出：

AttributeSchema	KeySchema
LocalSecondaryIndexSchema	
-----	-----

{ForumName}	{ForumName}
{}	

範例 2：在將 TableSchema 物件寫入管道之前，將新的金鑰和屬性定義項目新增至提供的 TableSchema 物件。根據預設，金鑰類型宣告為 'HASH'；使用值為 'RANGE' 的 -KeyType parameter 宣告範圍金鑰。TableSchema 物件也可以使用 -Schema 參數提供。

```
New-DDBTableSchema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
```

輸出：

AttributeSchema	KeySchema
LocalSecondaryIndexSchema	
-----	-----
{ForumName}	{ForumName}
{}	

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [Add-DDBKeySchema](#)。

ConvertFrom-DDBItem

以下程式碼範例顯示如何使用 ConvertFrom-DDBItem。

PowerShell V4 的工具

範例 1：ConvertFrom-DDBItem 用於將 Get-DDBItem 的結果從 DynamoDB AttributeValues 的雜湊轉換為字串和雙組等常見類型的雜湊。

```
@{
    SongTitle = 'Somewhere Down The Road'
    Artist    = 'No One You Know'
} | ConvertTo-DDBItem

Get-DDBItem -TableName 'Music' -Key $key | ConvertFrom-DDBItem
```

輸出：

Name	Value
----	-----
Genre	Country
Artist	No One You Know
Price	1.94

CriticRating	9
SongTitle	Somewhere Down The Road
AlbumTitle	Somewhat Famous

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ConvertFrom-DDBItem](#)。

ConvertTo-DDBItem

以下程式碼範例顯示如何使用 ConvertTo-DDBItem。

PowerShell V4 的工具

範例 1：將雜湊轉換為 DynamoDB 屬性值字典的範例。

```
@{
    SongTitle = 'Somewhere Down The Road'
    Artist    = 'No One You Know'
} | ConvertTo-DDBItem
```

Key	Value
---	----
SongTitle	Amazon.DynamoDBv2.Model.AttributeValue
Artist	Amazon.DynamoDBv2.Model.AttributeValue

範例 2：將雜湊轉換為 DynamoDB 屬性值字典的範例。

```
@{
    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
```

輸出：

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ConvertTo-DDBItem](#)。

Get-DDBBatchItem

以下程式碼範例顯示如何使用 Get-DDBBatchItem。

PowerShell V4 的工具

範例 1：從 DynamoDB 資料表 'Music' 和 'Songs' 取得具有 SongTitle "Somewhere Down The Road" 的項目。

```
$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
```

輸出：

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [BatchGetItem](#)。

Get-DDBItem

以下程式碼範例顯示如何使用 Get-DDBItem。

PowerShell V4 的工具

範例 1：使用分割區索引鍵 SongTitle 和排序索引鍵 Artist 傳回 DynamoDB 項目。

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem

Get-DDBItem -TableName 'Music' -Key $key | ConvertFrom-DDBItem
```

輸出：

Name	Value
----	-----

Genre	Country
SongTitle	Somewhere Down The Road
Price	1.94
Artist	No One You Know
CriticRating	9
AlbumTitle	Somewhat Famous

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [GetItem](#)。

Get-DDBTable

以下程式碼範例顯示如何使用 Get-DDBTable。

PowerShell V4 的工具

範例 1：傳回指定資料表的詳細資訊。

```
Get-DDBTable -TableName "myTable"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeTable](#)。

Get-DDBTableList

以下程式碼範例顯示如何使用 Get-DDBTableList。

PowerShell V4 的工具

範例 1：傳回所有資料表的詳細資訊，自動反覆運算，直到服務指出不存在其他資料表為止。

```
Get-DDBTableList
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListTables](#)。

Invoke-DDBQuery

以下程式碼範例顯示如何使用 Invoke-DDBQuery。

PowerShell V4 的工具

範例 1：叫用查詢，傳回具有指定 SongTitle 和 Artist 的 DynamoDB 項目。


```
$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
```

輸出：

Name	Value
----	-----
Genre	Country
Artist	No One You Know
Price	1.94
CriticRating	9
SongTitle	Somewhere Down The Road
AlbumTitle	Somewhat Famous

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的[查詢](#)。

Invoke-DDBScan

以下程式碼範例顯示如何使用 Invoke-DDBScan。

PowerShell V4 的工具

範例 1：傳回 Music 資料表中的所有項目。

```
Invoke-DDBScan -TableName 'Music' | ConvertFrom-DDBItem
```

輸出：

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

範例 2：傳回 Music 資料表中 CriticRating 大於或等於九的項目。

```
$scanFilter = @{
    CriticRating = [Amazon.DynamoDBv2.Model.Condition]{
        AttributeValueList = @( @{N = '9'} )
        ComparisonOperator = 'GE'
    }
}
Invoke-DDBScan -TableName 'Music' -ScanFilter $scanFilter | ConvertFrom-DDBItem
```

輸出：

Name	Value
----	-----
Genre	Country
Artist	No One You Know
Price	1.94
CriticRating	9
SongTitle	Somewhere Down The Road
AlbumTitle	Somewhat Famous

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 在 Cmdlet 參考 (V4) 中[掃描](#)。

New-DDBTable

以下程式碼範例顯示如何使用 New-DDBTable。

PowerShell V4 的工具

範例 1：此範例會建立名為 Thread 的資料表，其主索引鍵包含 'ForumName'（索引鍵類型雜湊）和 'Subject'（索引鍵類型範圍）。用來建構資料表的結構描述，可以使用 -Schema 參數，依照所示或指定方式輸送至每個 cmdlet。

```
$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
```

輸出：

```
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 : {}
```

範例 2：此範例會建立名為 Thread 的資料表，其主索引鍵包含 'ForumName'（索引鍵類型雜湊）和 'Subject'（索引鍵類型範圍）。也會定義本機次要索引。本機次要索引的索引鍵會從資料表 (ForumName) 上的主要雜湊索引鍵自動設定。用來建構資料表的結構描述，可以使用 -Schema 參數，依照所示或指定方式輸送至每個 cmdlet。

```
$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
```

輸出：

```
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}
```

範例 3：此範例示範如何使用單一管道建立名為 Thread 的資料表，其主要索引鍵包含 'ForumName'（索引鍵類型雜湊）和 'Subject'（索引鍵類型範圍）以及本機次要索引。如果管道或 -Schema 參數未提供新的 TableSchema 物件，Add-DDBKeySchema 和 Add-DDBIndexSchema 會為您建立新的 TableSchema 物件。TableSchema

```
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
```

輸出：

```
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}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateTable](#)。

New-DDBTableSchema

以下程式碼範例顯示如何使用 New-DDBTableSchema。

PowerShell V4 的工具

範例 1：建立空的 TableSchema 物件，準備好接受金鑰和索引定義，以用於建立新的 Amazon DynamoDB 資料表。傳回的物件可以輸送至 Add-DDBKeySchema、Add-DDBIndexSchema 和 New-DDBTable cmdlet，或使用每個 cmdlet 上的 -Schema 參數傳遞給它們。

```
New-DDBTableSchema
```

輸出：

AttributeSchema	KeySchema
LocalSecondaryIndexSchema	
-----	-----

{}	{}
{}	

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [New-DDBTableSchema](#)。

Remove-DDBItem

以下程式碼範例顯示如何使用 Remove-DDBItem。

PowerShell V4 的工具

範例 1：移除符合所提供金鑰的 DynamoDB 項目。

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem
Remove-DDBItem -TableName 'Music' -Key $key -Confirm:$false
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteItem](#)。

Remove-DDBTable

以下程式碼範例顯示如何使用 Remove-DDBTable。

PowerShell V4 的工具

範例 1：刪除指定的資料表。在操作繼續之前，系統會提示您進行確認。

```
Remove-DDBTable -TableName "myTable"
```

範例 2：刪除指定的資料表。在操作繼續之前，不會提示您進行確認。

```
Remove-DDBTable -TableName "myTable" -Force
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteTable](#)。

Set-DDBBatchItem

以下程式碼範例顯示如何使用 Set-DDBBatchItem。

PowerShell V4 的工具

範例 1：建立新的項目，或將現有項目取代為 DynamoDB 資料表中的新項目 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
```

輸出：

```
$requestItem = @{
    'Music' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
    'Songs' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
}

Set-DDBBatchItem -RequestItem $requestItem
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [BatchWriteItem](#)。

Set-DDBItem

以下程式碼範例顯示如何使用 Set-DDBItem。

PowerShell V4 的工具

範例 1：建立新項目，或將現有項目取代為新項目。

```
$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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [PutItem](#)。

Update-DDBItem

以下程式碼範例顯示如何使用 Update-DDBItem。

PowerShell V4 的工具

範例 1：使用分割區索引鍵 SongTitle 和排序索引鍵 Artist，將 DynamoDB 項目的類型屬性設定為 'Rap'。

```
$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
```

輸出：

Name	Value
----	-----
Genre	Rap

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UpdateItem](#)。

Update-DDBTable

以下程式碼範例顯示如何使用 Update-DDBTable。

PowerShell V4 的工具

範例 1：更新指定資料表的佈建輸送量。

```
Update-DDBTable -TableName "myTable" -ReadCapacity 10 -WriteCapacity 5
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [UpdateTable](#)。

使用 Tools for PowerShell V4 的 Amazon EC2 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 Amazon EC2 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Add-EC2CapacityReservation

以下程式碼範例顯示如何使用 Add-EC2CapacityReservation。

PowerShell V4 的工具

範例 1：此範例會使用指定的屬性建立新的容量保留

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

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateCapacityReservation](#)。

Add-EC2InternetGateway

以下程式碼範例顯示如何使用 Add-EC2InternetGateway。

PowerShell V4 的工具

範例 1：此範例會將指定的網際網路閘道連接至指定的 VPC。

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

範例 2：此範例會建立 VPC 和網際網路閘道，然後將網際網路閘道連接至 VPC。

```
$vpc = New-EC2Vpc -CidrBlock 10.0.0.0/16
```

```
New-EC2InternetGateway | Add-EC2InternetGateway -VpcId $vpc.VpcId
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [AttachInternetGateway](#)。

Add-EC2NetworkInterface

以下程式碼範例顯示如何使用 Add-EC2NetworkInterface。

PowerShell V4 的工具

範例 1：此範例會將指定的網路界面連接至指定的執行個體。

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

輸出：

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [AttachNetworkInterface](#)。

Add-EC2Volume

以下程式碼範例顯示如何使用 Add-EC2Volume。

PowerShell V4 的工具

範例 1：此範例會將指定的磁碟區連接至指定的執行個體，並使用指定的裝置名稱公開它。

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

輸出：

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

```
State           : attaching
VolumeId        : vol-12345678
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [AttachVolume](#)。

Add-EC2VpnGateway

以下程式碼範例顯示如何使用 Add-EC2VpnGateway。

PowerShell V4 的工具

範例 1：此範例會將指定的虛擬私有閘道連接到指定的 VPC。

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

輸出：

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [AttachVpnGateway](#)。

Approve-EC2VpcPeeringConnection

以下程式碼範例顯示如何使用 Approve-EC2VpcPeeringConnection。

PowerShell V4 的工具

範例 1：此範例會核准請求的 VpcPeeringConnectionId pcx-1dfad234b56ff78be

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

輸出：

```
AcceptorVpcInfo      : Amazon.EC2.Model.VpcPeeringConnectionVpcInfo
ExpirationTime       : 1/1/0001 12:00:00 AM
```

```
RequesterVpcInfo      : Amazon.EC2.Model.VpcPeeringConnectionVpcInfo
Status                : Amazon.EC2.Model.VpcPeeringConnectionStateReason
Tags                  : {}
VpcPeeringConnectionId : pcx-1dfad234b56ff78be
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [AcceptVpcPeeringConnection](#)。

Confirm-EC2ProductInstance

以下程式碼範例顯示如何使用 Confirm-EC2ProductInstance。

PowerShell V4 的工具

範例 1：此範例會判斷指定的產品程式碼是否與指定的執行個體相關聯。

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ConfirmProductInstance](#)。

Copy-EC2Image

以下程式碼範例顯示如何使用 Copy-EC2Image。

PowerShell V4 的工具

範例 1：此範例會將 'EU (愛爾蘭)' 區域中指定的 AMI 複製到 'US West (奧勒岡)' 區域。如果未指定 -Region，則會使用目前的預設區域做為目的地區域。

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

輸出：

```
ami-87654321
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CopyImage](#)。

Copy-EC2Snapshot

以下程式碼範例顯示如何使用 Copy-EC2Snapshot。

PowerShell V4 的工具

範例 1：此範例會將指定的快照從歐洲（愛爾蘭）區域複製到美國西部（奧勒岡）區域。

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

範例 2：如果您設定預設區域並省略區域參數，預設目的地區域即為預設區域。

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CopySnapshot](#)。

Deny-EC2VpcPeeringConnection

以下程式碼範例顯示如何使用 Deny-EC2VpcPeeringConnection。

PowerShell V4 的工具

範例 1：上述範例拒絕 VpcPeering 請求 ID pcx-01a2b3ce45fe67eb8 的請求

```
Deny-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-01a2b3ce45fe67eb8
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [RejectVpcPeeringConnection](#)。

Disable-EC2VgwRoutePropagation

以下程式碼範例顯示如何使用 Disable-EC2VgwRoutePropagation。

PowerShell V4 的工具

範例 1：此範例會停用 VGW 自動將路由傳播到指定的路由表。

```
Disable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DisableVgwRoutePropagation](#)。

Disable-EC2VpcClassicLink

以下程式碼範例顯示如何使用 Disable-EC2VpcClassicLink。

PowerShell V4 的工具

範例 1：此範例停用 vpc-01e23c4a5d6db78e9 的 EC2VpcClassicLink。它傳回 True 或 False

```
Disable-EC2VpcClassicLink -VpcId vpc-01e23c4a5d6db78e9
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DisableVpcClassicLink](#)。

Disable-EC2VpcClassicLinkDnsSupport

以下程式碼範例顯示如何使用 Disable-EC2VpcClassicLinkDnsSupport。

PowerShell V4 的工具

範例 1：此範例停用對 vpc-0b12d3456a7e8910d 的 ClassicLink DNS 支援

```
Disable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d
```

- 如需 API 詳細資訊，請參閱《Cmdlet 參考》中的 [DisableVpcClassicLinkDnsSupport](#) (V4)。AWS Tools for PowerShell

Dismount-EC2InternetGateway

以下程式碼範例顯示如何使用 Dismount-EC2InternetGateway。

PowerShell V4 的工具

範例 1：此範例會將指定的網際網路閘道與指定的 VPC 分離。

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

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DetachInternetGateway](#)。

Dismount-EC2NetworkInterface

以下程式碼範例顯示如何使用 Dismount-EC2NetworkInterface。

PowerShell V4 的工具

範例 1：此範例會移除網路界面與執行個體之間的指定附件。

```
Dismount-EC2NetworkInterface -AttachmentId eni-attach-1a2b3c4d -Force
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DetachNetworkInterface](#)。

Dismount-EC2Volume

以下程式碼範例顯示如何使用 Dismount-EC2Volume。

PowerShell V4 的工具

範例 1：此範例會分離指定的磁碟區。

```
Dismount-EC2Volume -VolumeId vol-12345678
```

輸出：

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

範例 2：您也可以指定執行個體 ID 和裝置名稱，以確保您分離的是正確的磁碟區。

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DetachVolume](#)。

Dismount-EC2VpnGateway

以下程式碼範例顯示如何使用 Dismount-EC2VpnGateway。

PowerShell V4 的工具

範例 1：此範例會將指定的虛擬私有閘道與指定的 VPC 分離。

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DetachVpnGateway](#)。

Edit-EC2CapacityReservation

以下程式碼範例顯示如何使用 Edit-EC2CapacityReservation。

PowerShell V4 的工具

範例 1：此範例透過將 instance 計數變更為 1 來修改 CapacityReservationId cr-0c1f2345db6f7cdba

```
Edit-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba -  
InstanceCount 1
```

輸出：

```
True
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ModifyCapacityReservation](#)。

Edit-EC2Host

以下程式碼範例顯示如何使用 Edit-EC2Host。

PowerShell V4 的工具

範例 1：此範例會將專用主機 h-01e23f4cd567890f3 的 AutoPlacement 設定修改為關閉

```
Edit-EC2Host -HostId h-03e09f8cd681609f3 -AutoPlacement off
```

輸出：

Successful	Unsuccessful
-----	-----
{h-01e23f4cd567890f3}	{}

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ModifyHosts](#)。

Edit-EC2IdFormat

以下程式碼範例顯示如何使用 Edit-EC2IdFormat。

PowerShell V4 的工具

範例 1：此範例會啟用指定資源類型的較長 ID 格式。

```
Edit-EC2IdFormat -Resource instance -UseLongId $true
```

範例 2：此範例會停用指定資源類型的較長 ID 格式。

```
Edit-EC2IdFormat -Resource instance -UseLongId $false
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ModifyIdFormat](#)。

Edit-EC2ImageAttribute

以下程式碼範例顯示如何使用 Edit-EC2ImageAttribute。

PowerShell V4 的工具

範例 1：此範例會更新指定 AMI 的描述。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Description "New description"
```

範例 2：此範例會公開 AMI（例如，讓 AWS 帳戶可以使用）。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType add -UserGroup all
```

範例 3：此範例會將 AMI 設為私有（例如，讓只有身為擁有者的您才能使用）。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType remove -UserGroup all
```

範例 4：此範例會將啟動許可授予指定的 AWS 帳戶。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType add -UserId 111122223333
```

範例 5：此範例會從指定的 移除啟動許可 AWS 帳戶。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType remove -UserId 111122223333
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ModifyImageAttribute](#)。

Edit-EC2InstanceAttribute

以下程式碼範例顯示如何使用 Edit-EC2InstanceAttribute。

PowerShell V4 的工具

範例 1：此範例會修改指定執行個體的執行個體類型。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceType m3.medium
```

範例 2：此範例透過指定「簡單」做為單一根 I/O 虛擬化 (SR-IOV) 網路支援參數 - SriovNetSupport..

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SriovNetSupport "simple"
```

範例 3：此範例會修改指定執行個體的安全群組。執行個體必須位於 VPC 中。您必須指定每個安全群組的 ID，而不是名稱。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -Group @( "sg-12345678",  
"sg-45678901" )
```

範例 4：此範例會啟用指定執行個體的 EBS I/O 最佳化。此功能不適用於所有執行個體類型。使用 EBS 最佳化執行個體時需支付額外的使用費。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -EbsOptimized $true
```

範例 5：此範例會啟用指定執行個體的來源/目的地檢查。若要讓 NAT 執行個體執行 NAT，值必須為 'false'。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SourceDestCheck $true
```

範例 6：此範例會停用指定執行個體的終止。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -DisableApiTermination $true
```

範例 7：此範例會變更指定的執行個體，以便在從執行個體啟動關機時終止。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceInitiatedShutdownBehavior  
terminate
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [ModifyInstanceAttribute](#)。

Edit-EC2InstanceCreditSpecification

以下程式碼範例顯示如何使用 Edit-EC2InstanceCreditSpecification。

PowerShell V4 的工具

範例 1：這會啟用執行個體 i-01234567890abcdef 的 T2 無限制額度。

```
$Credit = New-Object -TypeName Amazon.EC2.Model.InstanceCreditSpecificationRequest  
$Credit.InstanceId = "i-01234567890abcdef"
```

```
$Credit.CpuCredits = "unlimited"  
Edit-EC2InstanceCreditSpecification -InstanceCreditSpecification $Credit
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [ModifyInstanceCreditSpecification](#)。

Edit-EC2NetworkInterfaceAttribute

以下程式碼範例顯示如何使用 Edit-EC2NetworkInterfaceAttribute。

PowerShell V4 的工具

範例 1：此範例會修改指定的網路介面，以便在終止時刪除指定的附件。

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -  
Attachment_AttachmentId eni-attach-1a2b3c4d -Attachment_DeleteOnTermination $true
```

範例 2：此範例會修改指定網路介面的描述。

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Description "my  
description"
```

範例 3：此範例會修改指定網路介面的安全群組。

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Groups  
sg-1a2b3c4d
```

範例 4：此範例會停用指定網路介面的來源/目的地檢查。

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -SourceDestCheck  
$false
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [ModifyNetworkInterfaceAttribute](#)。

Edit-EC2ReservedInstance

以下程式碼範例顯示如何使用 Edit-EC2ReservedInstance。

PowerShell V4 的工具

範例 1：此範例會修改指定預留執行個體的可用區域、執行個體計數和平台。

```
$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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ModifyReservedInstances](#)。

Edit-EC2SnapshotAttribute

以下程式碼範例顯示如何使用 Edit-EC2SnapshotAttribute。

PowerShell V4 的工具

範例 1：此範例透過設定其 CreateVolumePermission 屬性，將指定的快照設為公有。

```
Edit-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
CreateVolumePermission -OperationType Add -GroupName all
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ModifySnapshotAttribute](#)。

Edit-EC2SpotFleetRequest

以下程式碼範例顯示如何使用 Edit-EC2SpotFleetRequest。

PowerShell V4 的工具

範例 1：此範例會更新指定 Spot 機群請求的目標容量。

```
Edit-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TargetCapacity 10
```

輸出：

```
True
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ModifySpotFleetRequest](#)。

Edit-EC2SubnetAttribute

以下程式碼範例顯示如何使用 Edit-EC2SubnetAttribute。

PowerShell V4 的工具

範例 1：此範例會啟用指定子網路的公有 IP 定址。

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $true
```

範例 2：此範例會停用指定子網路的公有 IP 定址。

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $false
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ModifySubnetAttribute](#)。

Edit-EC2VolumeAttribute

以下程式碼範例顯示如何使用 Edit-EC2VolumeAttribute。

PowerShell V4 的工具

範例 1：此範例會修改指定磁碟區的指定屬性。由於資料可能不一致而暫停之後，會自動恢復磁碟區的 I/O 操作。

```
Edit-EC2VolumeAttribute -VolumeId vol-12345678 -AutoEnableIO $true
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [ModifyVolumeAttribute](#)。

Edit-EC2VpcAttribute

以下程式碼範例顯示如何使用 Edit-EC2VpcAttribute。

PowerShell V4 的工具

範例 1：此範例支援指定 VPC 的 DNS 主機名稱。

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $true
```

範例 2：此範例停用對指定 VPC 的 DNS 主機名稱的支援。

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $false
```

範例 3：此範例支援指定 VPC 的 DNS 解析。

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $true
```

範例 4：此範例停用對指定 VPC DNS 解析的支援。

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $false
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ModifyVpcAttribute](#)。

Enable-EC2VgwRoutePropagation

以下程式碼範例顯示如何使用 Enable-EC2VgwRoutePropagation。

PowerShell V4 的工具

範例 1：此範例可讓指定的 VGW 自動將路由傳播至指定的路由表。

```
Enable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [EnableVgwRoutePropagation](#)。

Enable-EC2VolumeIO

以下程式碼範例顯示如何使用 Enable-EC2VolumeIO。

PowerShell V4 的工具

範例 1：如果停用 I/O 操作，此範例會啟用指定磁碟區的 I/O 操作。

```
Enable-EC2VolumeIO -VolumeId vol-12345678
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [EnableVolumeIO](#)。

Enable-EC2VpcClassicLink

以下程式碼範例顯示如何使用 Enable-EC2VpcClassicLink。

PowerShell V4 的工具

範例 1：此範例啟用 ClassicLink 的 VPC vpc-0123456b789b0d12f

```
Enable-EC2VpcClassicLink -VpcId vpc-0123456b789b0d12f
```

輸出：

```
True
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [EnableVpcClassicLink](#)。

Enable-EC2VpcClassicLinkDnsSupport

以下程式碼範例顯示如何使用 Enable-EC2VpcClassicLinkDnsSupport。

PowerShell V4 的工具

範例 1：此範例可讓 vpc-0b12d3456a7e8910d 支援 ClassicLink 的 DNS 主機名稱解析

```
Enable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```


- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [EnableVpcClassicLinkDnsSupport](#)。

Get-EC2AccountAttribute

以下程式碼範例顯示如何使用 Get-EC2AccountAttribute。

PowerShell V4 的工具

範例 1：此範例說明您是否可以在區域中的 EC2-Classic 和 EC2-VPC 中啟動執行個體，或僅在 EC2-VPC 中啟動執行個體。

```
(Get-EC2AccountAttribute -AttributeName supported-platforms).AttributeValues
```

輸出：

```
AttributeValue
-----
EC2
VPC
```

範例 2：此範例說明您的預設 VPC，如果區域中沒有預設 VPC，則為「無」。

```
(Get-EC2AccountAttribute -AttributeName default-vpc).AttributeValues
```

輸出：

```
AttributeValue
-----
vpc-12345678
```

範例 3：此範例說明您可以執行的隨需執行個體數量上限。

```
(Get-EC2AccountAttribute -AttributeName max-instances).AttributeValues
```

輸出：

```
AttributeValue
-----
```

20

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeAccountAttributes](#)。

Get-EC2Address

以下程式碼範例顯示如何使用 Get-EC2Address。

PowerShell V4 的工具

範例 1：此範例說明 EC2-Classic 中執行個體的指定彈性 IP 地址。

```
Get-EC2Address -AllocationId eipalloc-12345678
```

輸出：

```
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
```

範例 2：此範例說明 VPC 中執行個體的彈性 IP 地址。此語法需要 PowerShell 第 3 版或更新版本。

```
Get-EC2Address -Filter @{ Name="domain";Values="vpc" }
```

範例 3：此範例說明 EC2-Classic 中執行個體的指定彈性 IP 地址。

```
Get-EC2Address -PublicIp 203.0.113.17
```

輸出：

```
AllocationId      :
AssociationId     :
```

```

Domain           : standard
InstanceId        : i-12345678
NetworkInterfaceId :
NetworkInterfaceOwnerId :
PrivateIpAddress  :
PublicIp          : 203.0.113.17

```

範例 4：此範例說明 EC2-Classic 中執行個體的彈性 IP 地址。此語法需要 PowerShell 第 3 版或更新版本。

```
Get-EC2Address -Filter @{ Name="domain";Values="standard" }
```

範例 5：此範例說明所有彈性 IP 地址。

```
Get-EC2Address
```

範例 6：此範例會傳回篩選條件中提供的執行個體 ID 的公有和私有 IP

```
Get-EC2Address -Region eu-west-1 -Filter @{Name="instance-
id";Values="i-0c12d3f4f567ffb89"} | Select-Object PrivateIpAddress, PublicIp
```

輸出：

```

PrivateIpAddress PublicIp
-----
10.0.0.99         63.36.5.227

```

範例 7：此範例會擷取所有具有其配置 ID、關聯 ID 和執行個體 ID 的彈性 IPs

```
Get-EC2Address -Region eu-west-1 | Select-Object InstanceId, AssociationId,
AllocationId, PublicIp
```

輸出：

```

InstanceId        AssociationId        AllocationId        PublicIp
-----
17.212.120.178
i-0c123dfd3415bac67 eipassoc-0e123456bb7890bdb eipalloc-01cd23ebf45f7890c
17.212.124.77

```

```

eipalloc-012345678eeabcfad
17.212.225.7
i-0123d405c67e89a0c eipassoc-0c123b456783966ba eipalloc-0123cdd456a8f7892
37.216.52.173
i-0f1bf2f34c5678d09 eipassoc-0e12934568a952d96 eipalloc-0e1c23e4d5e6789e4
37.218.222.278
i-012e3cb4df567e8aa eipassoc-0d1b2fa4d67d03810 eipalloc-0123f456f78a01b58
37.210.82.27
i-0123bcf4b567890e1 eipassoc-01d2345f678903fb1 eipalloc-0e1db23cfef5c45c7
37.215.222.270

```

範例 8：此範例會擷取值為 'Prod' 且符合標籤索引鍵 'Category' 的 EC2 IP 地址清單

```
Get-EC2Address -Filter @{"Name"="tag:Category";Values="Prod"}
```

輸出：

```

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}

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeAddresses](#)。

Get-EC2AvailabilityZone

以下程式碼範例顯示如何使用 Get-EC2AvailabilityZone。

PowerShell V4 的工具

範例 1：此範例說明目前可用區域的可用區域。

```
Get-EC2AvailabilityZone
```

輸出：

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

範例 2：此範例說明處於受損狀態的任何可用區域。此範例使用的語法需要 PowerShell 第 3 版或更新版本。

```
Get-EC2AvailabilityZone -Filter @{ Name="state";Values="impaired" }
```

範例 3：使用 PowerShell 第 2 版時，您必須使用 New-Object 來建立篩選條件。

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = "impaired"

Get-EC2AvailabilityZone -Filter $filter
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeAvailabilityZones](#)。

Get-EC2BundleTask

以下程式碼範例顯示如何使用 Get-EC2BundleTask。

PowerShell V4 的工具

範例 1：此範例說明指定的套件任務。

```
Get-EC2BundleTask -BundleId bun-12345678
```

範例 2：此範例說明狀態為「完成」或「失敗」的套件任務。

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
```

```
$filter.Values = @( "complete", "failed" )  
  
Get-EC2BundleTask -Filter $filter
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeBundleTasks](#)。

Get-EC2CapacityReservation

以下程式碼範例顯示如何使用 Get-EC2CapacityReservation。

PowerShell V4 的工具

範例 1：此範例說明區域的一或多個容量保留

```
Get-EC2CapacityReservation -Region eu-west-1
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeCapacityReservations](#)。

Get-EC2ConsoleOutput

以下程式碼範例顯示如何使用 Get-EC2ConsoleOutput。

PowerShell V4 的工具

範例 1：此範例會取得指定 Linux 執行個體的主控制台輸出。主控台輸出已編碼。

```
Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456
```

輸出：

InstanceId	Output
-----	-----
i-0e194d3c47c123637	WyAgICAwLjAwMDAwMF0gQ29tbW...bGU9dHR5UzAgc2Vs

範例 2：此範例會將編碼的主控制台輸出存放在變數中，然後將其解碼。

```
$Output_encoded = (Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456).Output
[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($Output_encoded))
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetConsoleOutput](#)。

Get-EC2CustomerGateway

以下程式碼範例顯示如何使用 Get-EC2CustomerGateway。

PowerShell V4 的工具

範例 1：此範例說明指定的客戶閘道。

```
Get-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

輸出：

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

範例 2：此範例說明狀態為待定或可用的任何客戶閘道。

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2CustomerGateway -Filter $filter
```

範例 3：此範例說明您的所有客戶閘道。

```
Get-EC2CustomerGateway
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeCustomerGateways](#)。

Get-EC2DhcpOption

以下程式碼範例顯示如何使用 Get-EC2DhcpOption。

PowerShell V4 的工具

範例 1：此範例列出您的 DHCP 選項集。

```
Get-EC2DhcpOption
```

輸出：

DhcpConfigurations	DhcpOptionsId	Tag
-----	-----	---
{domain-name, domain-name-servers}	dopt-1a2b3c4d	{}
{domain-name, domain-name-servers}	dopt-2a3b4c5d	{}
{domain-name-servers}	dopt-3a4b5c6d	{}

範例 2：此範例會取得指定 DHCP 選項集的組態詳細資訊。

```
(Get-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d).DhcpConfigurations
```

輸出：

Key	Values
---	-----


```
domain-name           {abc.local}
domain-name-servers   {10.0.0.101, 10.0.0.102}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeDhcpOptions](#)。

Get-EC2FlowLog

以下程式碼範例顯示如何使用 Get-EC2FlowLog。

PowerShell V4 的工具

範例 1：此範例說明日誌目的地類型為 's3' 的一或多個流程日誌

```
Get-EC2FlowLog -Filter @{"Name="log-destination-type";Values="s3"}
```

輸出：

```
CreationTime           : 2/25/2019 9:07:36 PM
DeliverLogsErrorMessage :
DeliverLogsPermissionArn :
DeliverLogsStatus      : SUCCESS
FlowLogId              : fl-01b2e3d45f67f8901
FlowLogStatus          : ACTIVE
LogDestination         : arn:aws:s3:::amzn-s3-demo-bucket-dd-tata
LogDestinationType     : s3
LogGroupName           :
ResourceId             : eni-01d2dda3456b7e890
TrafficType            : ALL
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeFlowLogs](#)。

Get-EC2Host

以下程式碼範例顯示如何使用 Get-EC2Host。

PowerShell V4 的工具

範例 1：此範例會傳回 EC2 主機詳細資訊

Get-EC2Host

輸出：

```
AllocationTime      : 3/23/2019 4:55:22 PM
AutoPlacement       : off
AvailabilityZone     : eu-west-1b
AvailableCapacity   : Amazon.EC2.Model.AvailableCapacity
ClientToken         :
HostId              : h-01e23f4cd567890f1
HostProperties       : Amazon.EC2.Model.HostProperties
HostReservationId   :
Instances           : {}
ReleaseTime         : 1/1/0001 12:00:00 AM
State               : available
Tags                : {}
```

範例 2：此範例會查詢主機 h-01e23f4cd567899f1 的 AvailableInstanceCapacity

```
Get-EC2Host -HostId h-01e23f4cd567899f1 | Select-Object -ExpandProperty
AvailableCapacity | Select-Object -expand AvailableInstanceCapacity
```

輸出：

```
AvailableCapacity InstanceType TotalCapacity
-----
11                m4.xlarge      11
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeHosts](#)。

Get-EC2HostReservationOffering

以下程式碼範例顯示如何使用 Get-EC2HostReservationOffering。

PowerShell V4 的工具

範例 1：此範例說明專用主機保留，可供購買給指定篩選條件 'instance-family'，其中 PaymentOption 為 'NoUpfront'

```
Get-EC2HostReservationOffering -Filter @{Name="instance-family";Values="m4"} |
Where-Object PaymentOption -eq NoUpfront
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeHostReservationOfferings](#)。

Get-EC2HostReservationPurchasePreview

以下程式碼範例顯示如何使用 Get-EC2HostReservationPurchasePreview。

PowerShell V4 的工具

範例 1：此範例會預覽具有符合專用主機 h-01e23f4cd567890f1 之組態的保留購買

```
Get-EC2HostReservationPurchasePreview -OfferingId hro-0c1f23456789d0ab -HostIdSet
h-01e23f4cd567890f1
```

輸出：

```
CurrencyCode Purchase TotalHourlyPrice TotalUpfrontPrice
-----
{}           1.307           0.000
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetHostReservationPurchasePreview](#)。

Get-EC2IdFormat

以下程式碼範例顯示如何使用 Get-EC2IdFormat。

PowerShell V4 的工具

範例 1：此範例說明指定資源類型的 ID 格式。

```
Get-EC2IdFormat -Resource instance
```

輸出：

Resource	UseLongIds
-----	-----
instance	False

範例 2：此範例說明支援較長 ID 的所有資源類型的 IDs 格式。

```
Get-EC2IdFormat
```

輸出：

Resource	UseLongIds
-----	-----
reservation	False
instance	False

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeIdFormat](#)。

Get-EC2IdentityIdFormat

以下程式碼範例顯示如何使用 Get-EC2IdentityIdFormat。

PowerShell V4 的工具

範例 1：此範例會傳回指定角色之 resource 'image' 的 ID 格式

```
Get-EC2IdentityIdFormat -PrincipalArn arn:aws:iam::123456789511:role/JDBC -Resource
image
```

輸出：

```
Deadline          Resource UseLongIds
-----
8/2/2018 11:30:00 PM image      True
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeIdentityIdFormat](#)。

Get-EC2Image

以下程式碼範例顯示如何使用 Get-EC2Image。

PowerShell V4 的工具

範例 1：此範例說明指定的 AMI。

```
Get-EC2Image -ImageId ami-12345678
```

輸出：

```
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
```

範例 2：此範例說明您擁有 AMIs。

```
Get-EC2Image -owner self
```

範例 3：此範例說明執行 Microsoft Windows Server 的公 AMIs。

```
Get-EC2Image -Filter @{ Name="platform"; Values="windows" }
```

範例 4：此範例說明 'us-west-2' 區域中的所有公有 AMIs。

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeImages](#)。

Get-EC2ImageAttribute

以下程式碼範例顯示如何使用 Get-EC2ImageAttribute。

PowerShell V4 的工具

範例 1：此範例會取得指定 AMI 的描述。

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute description
```

輸出：

```
BlockDeviceMappings : {}
Description           : My image description
ImageId               : ami-12345678
KernelId              :
LaunchPermissions     : {}
```

```
ProductCodes      : {}  
RamdiskId         :  
SriovNetSupport   :
```

範例 2：此範例取得指定 AMI 的啟動許可。

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

輸出：

```
BlockDeviceMappings : {}  
Description          :  
ImageId              : ami-12345678  
KernelId             :  
LaunchPermissions    : {all}  
ProductCodes         : {}  
RamdiskId            :  
SriovNetSupport      :
```

範例 3：此範例測試是否已啟用增強型聯網。

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute sriovNetSupport
```

輸出：

```
BlockDeviceMappings : {}  
Description          :  
ImageId              : ami-12345678  
KernelId             :  
LaunchPermissions    : {}  
ProductCodes         : {}  
RamdiskId            :  
SriovNetSupport      : simple
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeImageAttribute](#)。

Get-EC2ImageByName

以下程式碼範例顯示如何使用 Get-EC2ImageByName。

PowerShell V4 的工具

範例 1：此範例說明目前支援的完整一組篩選條件名稱。

```
Get-EC2ImageByName
```

輸出：

```
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
```

範例 2：此範例說明指定的 AMI。使用此命令尋找 AMI 很有幫助，因為 每個月都會 AWS 發行具有最新更新的 Windows AMIs。您可以將「ImageId」指定為 New-EC2Instance，以使用指定篩選條件的目前 AMI 啟動執行個體。

```
Get-EC2ImageByName -Names WINDOWS_2016_BASE
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [Get-EC2ImageByName](#)。

Get-EC2ImportImageTask

以下程式碼範例顯示如何使用 Get-EC2ImportImageTask。

PowerShell V4 的工具

範例 1：此範例說明指定的映像匯入任務。

```
Get-EC2ImportImageTask -ImportTaskId import-ami-hgfedcba
```

輸出：

```
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     :
```

範例 2：此範例說明所有映像匯入任務。

```
Get-EC2ImportImageTask
```

輸出：

```
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     :

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeImportImageTasks](#)。

Get-EC2ImportSnapshotTask

以下程式碼範例顯示如何使用 Get-EC2ImportSnapshotTask。

PowerShell V4 的工具

範例 1：此範例說明指定的快照匯入任務。

```
Get-EC2ImportSnapshotTask -ImportTaskId import-snap-abcdefgh
```

輸出：

Description	ImportTaskId	SnapshotTaskDetail
-----	-----	-----
Disk Image Import 1	import-snap-abcdefgh	
Amazon.EC2.Model.SnapshotTaskDetail		

範例 2：此範例說明所有快照匯入任務。

```
Get-EC2ImportSnapshotTask
```

輸出：

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeImportSnapshotTasks](#)。

Get-EC2Instance

以下程式碼範例顯示如何使用 Get-EC2Instance。

PowerShell V4 的工具

範例 1：此範例說明指定的執行個體。

```
(Get-EC2Instance -InstanceId i-12345678).Instances
```

輸出：

```
AmiLaunchIndex      : 0
Architecture        : x86_64
BlockDeviceMappings : {/dev/sda1}
ClientToken          : TleEy1448154045270
EbsOptimized         : False
Hypervisor           : xen
IamInstanceProfile   : Amazon.EC2.Model.IamInstanceProfile
ImageId              : ami-12345678
InstanceId            : i-12345678
InstanceLifecycle    :
InstanceType         : t2.micro
KernelId             :
KeyName              : my-key-pair
LaunchTime           : 12/4/2015 4:44:40 PM
Monitoring           : Amazon.EC2.Model.Monitoring
NetworkInterfaces    : {ip-10-0-2-172.us-west-2.compute.internal}
Placement            : Amazon.EC2.Model.Placement
```

```

Platform           : Windows
PrivateDnsName     : ip-10-0-2-172.us-west-2.compute.internal
PrivateIpAddress   : 10.0.2.172
ProductCodes       : {}
PublicDnsName      :
PublicIpAddress    :
RamdiskId          :
RootDeviceName     : /dev/sda1
RootDeviceType     : ebs
SecurityGroups     : {default}
SourceDestCheck    : True
SpotInstanceRequestId :
SriovNetSupport    :
State              : Amazon.EC2.Model.InstanceState
StateReason        :
StateTransitionReason :
SubnetId           : subnet-12345678
Tags               : {Name}
VirtualizationType : hvm
VpcId              : vpc-12345678

```

範例 2：此範例說明目前區域中所有執行個體，依保留分組。若要查看執行個體詳細資訊，請在每個保留物件中展開執行個體集合。

```
Get-EC2Instance
```

輸出：

```

GroupNames      : {}
Groups          : {}
Instances       : {}
OwnerId         : 123456789012
RequesterId     : 226008221399
ReservationId   : r-c5df370c

GroupNames      : {}
Groups          : {}
Instances       : {}
OwnerId         : 123456789012
RequesterId     : 854251627541
ReservationId   : r-63e65bab
...

```

範例 3：此範例說明使用篩選條件來查詢 VPC 特定子網路中的 EC2 執行個體。

```
(Get-EC2Instance -Filter @{{Name="vpc-id";Values="vpc-1a2bc34d"},@{Name="subnet-id";Values="subnet-1a2b3c4d"}}).Instances
```

輸出：

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			

範例 4：此範例說明使用具有多個值的篩選條件來查詢正在執行和已停止的 EC2 執行個體

```
$InstanceParams = @{
    Filter = @(
        @{'Name' = 'instance-state-name';'Values' = @("running","stopped")}
    )
}

(Get-EC2Instance @InstanceParams).Instances
```

輸出：

InstanceId	InstanceType	Platform	PrivateIpAddress	PublicIpAddress
SecurityGroups	SubnetId	VpcId		
i-05a9...f6c5f46e18	t3.medium		10.0.1.7	...
subnet-1a2b3c4d	vpc-1a2b3c4d			
i-02cf...945c4fdd07	t3.medium	Windows	10.0.1.8	...
subnet-1a2b3c4d	vpc-1a2b3c4d			
i-0ac0...c037f9f3a1	t3.xlarge	Windows	10.0.1.10	...
subnet-1a2b3c4d	vpc-1a2b3c4d			
i-066b...57b7b08888	t3.medium	Windows	10.0.1.11	...
subnet-1a2b3c4d	vpc-1a2b3c4d			
i-0fee...82e83ccd72	t3.medium	Windows	10.0.1.5	...
subnet-1a2b3c4d	vpc-1a2b3c4d			

```
i-0a68...274cc5043b t3.medium    Windows  10.0.1.6    ...
subnet-1a2b3c4d vpc-1a2b3c4d
```

範例 5：此範例說明使用具有多個值的篩選條件來查詢正在執行和停止的 EC2 執行個體，以及使用 `Select-Object` cmdlet 選擇要輸出的特定值。

```
$InstanceParams = @{
    Filter = @(
        @{'Name' = 'instance-state-name'; 'Values' = @("running","stopped")}
    )
}

$SelectParams = @{
    Property = @(
        "InstanceId", "InstanceType", "Platform", "PrivateIpAddress",
        @{'Name'="Name";Expression={$_.Tags[$_].Tags.Key.IndexOf("Name")}.Value}},
        @{'Name'="State";Expression={$_.State.Name}}
    )
}

$result = Get-EC2Instance @InstanceParams
$result.Instances | Select-Object @SelectParams | Format-Table -AutoSize
```

輸出：

InstanceId	InstanceType	Platform	PrivateIpAddress	Name	State
-----	-----	-----	-----	----	----
i-05a9...f6c5f46e18	t3.medium		10.0.1.7	ec2-name-01	running
i-02cf...945c4fdd07	t3.medium	Windows	10.0.1.8	ec2-name-02	stopped
i-0ac0...c037f9f3a1	t3.xlarge	Windows	10.0.1.10	ec2-name-03	running
i-066b...57b7b08888	t3.medium	Windows	10.0.1.11	ec2-name-04	stopped
i-0fee...82e83ccd72	t3.medium	Windows	10.0.1.5	ec2-name-05	running
i-0a68...274cc5043b	t3.medium	Windows	10.0.1.6	ec2-name-06	stopped

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeInstances](#)。

Get-EC2InstanceAttribute

以下程式碼範例顯示如何使用 `Get-EC2InstanceAttribute`。

PowerShell V4 的工具

範例 1：此範例說明指定執行個體的執行個體類型。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute instanceType
```

輸出：

```
InstanceType           : t2.micro
```

範例 2：此範例說明是否為指定的執行個體啟用增強型聯網。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

輸出：

```
SriovNetSupport        : simple
```

範例 3：此範例說明指定執行個體的安全群組。

```
(Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute groupSet).Groups
```

輸出：

```
GroupId
-----
sg-12345678
sg-45678901
```

範例 4：此範例說明指定執行個體是否啟用 EBS 最佳化。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

輸出：

```
EbsOptimized           : False
```

範例 5：此範例說明指定執行個體的 'disableApiTermination' 屬性。


```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

輸出：

```
DisableApiTermination          : False
```

範例 6：此範例說明指定執行個體的 'instanceInitiatedShutdownBehavior' 屬性。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute  
instanceInitiatedShutdownBehavior
```

輸出：

```
InstanceInitiatedShutdownBehavior : stop
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeInstanceAttribute](#)。

Get-EC2InstanceMetadata

以下程式碼範例顯示如何使用 Get-EC2InstanceMetadata。

PowerShell V4 的工具

範例 1：列出可查詢的可用執行個體中繼資料類別。

```
Get-EC2InstanceMetadata -ListCategory
```

輸出：

```
AmiId  
LaunchIndex  
ManifestPath  
AncestorAmiId  
BlockDeviceMapping  
InstanceId  
InstanceType  
LocalHostname  
LocalIpv4  
KernelId
```

```
AvailabilityZone
ProductCode
PublicHostname
PublicIpv4
PublicKey
RamdiskId
Region
ReservationId
SecurityGroup
UserData
InstanceMonitoring
IdentityDocument
IdentitySignature
IdentityPkcs7
```

範例 2：傳回用來啟動執行個體的 Amazon Machine Image (AMI) ID。

```
Get-EC2InstanceMetadata -Category AmiId
```

輸出：

```
ami-b2e756ca
```

範例 3：此範例會查詢執行個體的 JSON 格式身分文件。

```
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"
```

```
}
```

範例 4：此範例使用路徑查詢來取得執行個體的網路界面 mac。

```
Get-EC2InstanceMetadata -Path "/network/interfaces/macs"
```

輸出：

```
02:80:7f:ef:4c:e0/
```

範例 5：如果有與執行個體相關聯的 IAM 角色，會傳回執行個體設定檔上次更新的相關資訊，包括執行個體的 LastUpdated date、InstanceProfileArn 和 InstanceProfileId。

```
Get-EC2InstanceMetadata -Path "/iam/info"
```

輸出：

```
{
  "Code" : "Success",
  "LastUpdated" : "2018-03-08T03:38:40Z",
  "InstanceProfileArn" : "arn:aws:iam::111122223333:instance-profile/
MyLaunchRole_Profile",
  "InstanceProfileId" : "AIPAI4...WVK2RW"
}
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [Get-EC2InstanceMetadata](#)。

Get-EC2InstanceStatus

以下程式碼範例顯示如何使用 Get-EC2InstanceStatus。

PowerShell V4 的工具

範例 1：此範例說明指定執行個體的狀態。

```
Get-EC2InstanceStatus -InstanceId i-12345678
```

輸出：

```
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
```

輸出：

Code	Name
----	----
16	running

```
$status.Status
```

輸出：

Details	Status
-----	-----
{reachability}	ok

```
$status.SystemStatus
```

輸出：

Details	Status
-----	-----
{reachability}	ok

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeInstanceStatus](#)。

Get-EC2InternetGateway

以下程式碼範例顯示如何使用 Get-EC2InternetGateway。

PowerShell V4 的工具

範例 1：此範例說明指定的網際網路閘道。

```
Get-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

輸出：

Attachments	InternetGatewayId	Tags
-----	-----	----
{vpc-1a2b3c4d}	igw-1a2b3c4d	{}

範例 2：此範例說明您的所有網際網路閘道。

```
Get-EC2InternetGateway
```

輸出：

Attachments	InternetGatewayId	Tags
-----	-----	----
{vpc-1a2b3c4d}	igw-1a2b3c4d	{}
{}	igw-2a3b4c5d	{}

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeInternetGateways](#)。

Get-EC2KeyPair

以下程式碼範例顯示如何使用 Get-EC2KeyPair。

PowerShell V4 的工具

範例 1：此範例說明指定的金鑰對。

```
Get-EC2KeyPair -KeyName my-key-pair
```

輸出：

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
```

範例 2：此範例說明您的所有金鑰對。

```
Get-EC2KeyPair
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeKeyPairs](#)。

Get-EC2NetworkAcl

以下程式碼範例顯示如何使用 Get-EC2NetworkAcl。

PowerShell V4 的工具

範例 1：此範例說明指定的網路 ACL。

```
Get-EC2NetworkAcl -NetworkAclId acl-12345678
```

輸出：

```
Associations : {aclassoc-1a2b3c4d}
Entries      : {Amazon.EC2.Model.NetworkAclEntry, Amazon.EC2.Model.NetworkAclEntry}
IsDefault    : False
NetworkAclId : acl-12345678
Tags         : {Name}
VpcId        : vpc-12345678
```

範例 2：此範例說明指定網路 ACL 的規則。

```
(Get-EC2NetworkAcl -NetworkAclId acl-12345678).Entries
```

輸出：

```
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

```

範例 3：此範例說明您的所有網路 ACLs。

```
Get-EC2NetworkAcl
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeNetworkAcls](#)。

Get-EC2NetworkInterface

以下程式碼範例顯示如何使用 Get-EC2NetworkInterface。

PowerShell V4 的工具

範例 1：此範例說明指定的網路介面。

```
Get-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

輸出：

```

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
```

範例 2：此範例說明您的所有網路介面。

```
Get-EC2NetworkInterface
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeNetworkInterfaces](#)。

Get-EC2NetworkInterfaceAttribute

以下程式碼範例顯示如何使用 Get-EC2NetworkInterfaceAttribute。

PowerShell V4 的工具

範例 1：此範例說明指定的網路介面。

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
Attachment
```

輸出：

```
Attachment          : Amazon.EC2.Model.NetworkInterfaceAttachment
```

範例 2：此範例說明指定的網路介面。

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
Description
```

輸出：

```
Description         : My description
```

範例 3：此範例說明指定的網路介面。

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
GroupSet
```


輸出：

```
Groups           : {my-security-group}
```

範例 4：此範例說明指定的網路界面。

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute  
SourceDestCheck
```

輸出：

```
SourceDestCheck   : True
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeNetworkInterfaceAttribute](#)。

Get-EC2PasswordData

以下程式碼範例顯示如何使用 Get-EC2PasswordData。

PowerShell V4 的工具

範例 1：此範例會解密 Amazon EC2 指派給指定 Windows 執行個體管理員帳戶的密碼。指定 pem 檔案時，會自動假設 -Decrypt 切換的設定。

```
Get-EC2PasswordData -InstanceId i-12345678 -PemFile C:\path\my-key-pair.pem
```

輸出：

```
mYZ(PA9?C)Q
```

範例 2：(僅限 Windows PowerShell) 檢查執行個體，以判斷用來啟動執行個體的金鑰對名稱，然後嘗試在 AWS Toolkit for Visual Studio 的組態存放區中尋找對應的金鑰對資料。如果找到金鑰對資料，則會解密密碼。

```
Get-EC2PasswordData -InstanceId i-12345678 -Decrypt
```

輸出：

```
mYZ(PA9?C)Q
```

範例 3：傳回執行個體的加密密碼資料。

```
Get-EC2PasswordData -InstanceId i-12345678
```

輸出：

```
iVz3BAK/WAXV.....dqt8WeMA==
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetPasswordData](#)。

Get-EC2PlacementGroup

以下程式碼範例顯示如何使用 Get-EC2PlacementGroup。

PowerShell V4 的工具

範例 1：此範例說明指定的置放群組。

```
Get-EC2PlacementGroup -GroupName my-placement-group
```

輸出：

GroupName	State	Strategy
-----	-----	-----
my-placement-group	available	cluster

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribePlacementGroups](#)。

Get-EC2PrefixList

以下程式碼範例顯示如何使用 Get-EC2PrefixList。

PowerShell V4 的工具

範例 1：此範例會擷取區域字首清單格式 AWS 服務的可用。

```
Get-EC2PrefixList
```

輸出：

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribePrefixLists](#)。

Get-EC2Region

以下程式碼範例顯示如何使用 Get-EC2Region。

PowerShell V4 的工具

範例 1：此範例說明可供您使用的區域。

```
Get-EC2Region
```

輸出：

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeRegions](#)。

Get-EC2RouteTable

以下程式碼範例顯示如何使用 Get-EC2RouteTable。

PowerShell V4 的工具

範例 1：此範例說明所有路由表。

```
Get-EC2RouteTable
```

輸出：

```
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  :
```

範例 2：此範例會傳回指定路由表的詳細資訊。

```
Get-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

範例 3：此範例說明指定 VPC 的路由表。

```
Get-EC2RouteTable -Filter @{ Name="vpc-id"; Values="vpc-1a2b3c4d" }
```

輸出：

```
Associations      : {rtbassoc-12345678}
PropagatingVgws   : {}
Routes            : {, }
RouteTableId      : rtb-1a2b3c4d
Tags              : {}
VpcId             : vpc-1a2b3c4d
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeRouteTables](#)。

Get-EC2ScheduledInstance

以下程式碼範例顯示如何使用 Get-EC2ScheduledInstance。

PowerShell V4 的工具

範例 1：此範例說明指定的排程執行個體。

```
Get-EC2ScheduledInstance -ScheduledInstanceId sci-1234-1234-1234-1234-123456789012
```

輸出：

```
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
```

範例 2：此範例說明所有已排程的執行個體。

```
Get-EC2ScheduledInstance
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeScheduledInstances](#)。

Get-EC2ScheduledInstanceAvailability

以下程式碼範例顯示如何使用 Get-EC2ScheduledInstanceAvailability。

PowerShell V4 的工具

範例 1：此範例描述從指定日期開始，每週週日發生的排程。

```
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
```

輸出：

```
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
...
```

範例 2：若要縮小結果範圍，您可以為作業系統、網路和執行個體類型等條件新增篩選條件。

```
-Filter @{ Name="platform";Values="Linux/UNIX" },@{ Name="network-
platform";Values="EC2-VPC" },@{ Name="instance-type";Values="c4.large" }
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeScheduledInstanceAvailability](#)。

Get-EC2SecurityGroup

以下程式碼範例顯示如何使用 Get-EC2SecurityGroup。

PowerShell V4 的工具

範例 1：此範例說明 VPC 的指定安全群組。使用屬於 VPC 的安全群組時，您必須使用安全群組 ID (-GroupId 參數)，而非名稱 (-GroupName 參數) 來參考群組。

```
Get-EC2SecurityGroup -GroupId sg-12345678
```

輸出：

```
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
```

範例 2：此範例說明 EC2-Classic 的指定安全群組。使用 EC2-Classic 的安全群組時，您可以使用群組名稱 (-GroupName 參數) 或群組 ID (-GroupId 參數) 來參考安全群組。

```
Get-EC2SecurityGroup -GroupName my-security-group
```

輸出：

```
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           :
```

範例 3：此範例會擷取 vpc-0fc1ff23456b789eb 的所有安全群組

```
Get-EC2SecurityGroup -Filter @{"Name"="vpc-id";Values="vpc-0fc1ff23456b789eb"}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeSecurityGroups](#)。

Get-EC2Snapshot

以下程式碼範例顯示如何使用 Get-EC2Snapshot。

PowerShell V4 的工具

範例 1：此範例說明指定的快照。

```
Get-EC2Snapshot -SnapshotId snap-12345678
```

輸出：

```
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
```

範例 2：此範例說明具有 'Name' 標籤的快照。

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" }
```

範例 3：此範例說明具有值為 'TestValue' 之 'Name' 標籤的快照。

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" -and
                        $_.Tags.Value -eq "TestValue" }
```

範例 4：此範例說明您的所有快照。


```
Get-EC2Snapshot -Owner self
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeSnapshots](#)。

Get-EC2SnapshotAttribute

以下程式碼範例顯示如何使用 Get-EC2SnapshotAttribute。

PowerShell V4 的工具

範例 1：此範例說明指定快照的指定屬性。

```
Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute ProductCodes
```

輸出：

CreateVolumePermissions	ProductCodes	SnapshotId
-----	-----	-----
{}	{}	snap-12345678

範例 2：此範例說明指定快照的指定屬性。

```
(Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute  
CreateVolumePermission).CreateVolumePermissions
```

輸出：

Group	UserId
-----	-----
all	

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeSnapshotAttribute](#)。

Get-EC2SpotDatafeedSubscription

以下程式碼範例顯示如何使用 Get-EC2SpotDatafeedSubscription。

PowerShell V4 的工具

範例 1：此範例說明您的 Spot 執行個體資料饋送。

```
Get-EC2SpotDatafeedSubscription
```

輸出：

```
Bucket   : amzn-s3-demo-bucket
Fault    :
OwnerId  : 123456789012
Prefix   : spotdata
State    : Active
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeSpotDatafeedSubscription](#)。

Get-EC2SpotFleetInstance

以下程式碼範例顯示如何使用 Get-EC2SpotFleetInstance。

PowerShell V4 的工具

範例 1：此範例說明與指定 Spot 機群請求相關聯的執行個體。

```
Get-EC2SpotFleetInstance -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
```

輸出：

InstanceId	InstanceType	SpotInstanceRequestId
-----	-----	-----
i-f089262a	c3.large	sir-12345678
i-7e8b24a4	c3.large	sir-87654321

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeSpotFleetInstances](#)。

Get-EC2SpotFleetRequest

以下程式碼範例顯示如何使用 Get-EC2SpotFleetRequest。

PowerShell V4 的工具

範例 1：此範例說明指定的 Spot 機群請求。

```
Get-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE  
| format-list
```

輸出：

```
ConfigData           : Amazon.EC2.Model.SpotFleetRequestConfigData  
CreateTime           : 12/26/2015 8:23:33 AM  
SpotFleetRequestId   : sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE  
SpotFleetRequestState : active
```

範例 2：此範例說明所有 Spot 機群請求。

```
Get-EC2SpotFleetRequest
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeSpotFleetRequests](#)。

Get-EC2SpotFleetRequestHistory

以下程式碼範例顯示如何使用 Get-EC2SpotFleetRequestHistory。

PowerShell V4 的工具

範例 1：此範例說明指定 Spot 機群請求的歷史記錄。

```
Get-EC2SpotFleetRequestHistory -SpotFleetRequestId sfr-73fbd2ce-  
aa30-494c-8788-1cee4EXAMPLE -StartTime 2015-12-26T00:00:00Z
```

輸出：

```
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
```

輸出：

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeSpotFleetRequestHistory](#)。

Get-EC2SpotInstanceRequest

以下程式碼範例顯示如何使用 Get-EC2SpotInstanceRequest。

PowerShell V4 的工具

範例 1：此範例說明指定的 Spot 執行個體請求。

```
Get-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

輸出：

```
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
```

範例 2：此範例說明所有 Spot 執行個體請求。

```
Get-EC2SpotInstanceRequest
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeSpotInstanceRequests](#)。

Get-EC2SpotPriceHistory

以下程式碼範例顯示如何使用 Get-EC2SpotPriceHistory。

PowerShell V4 的工具

範例 1：此範例會取得指定執行個體類型和可用區域的 Spot 價格歷史記錄中最後 10 個項目。請注意，針對 -AvailabilityZone 參數指定的值，對於提供給 cmdlet 的 -Region 參數（未顯示於範例中）或在 shell 中設定為預設值的區域值，必須是有效的。此範例命令假設已在環境中設定 'us-west-2' 的預設區域。

```
Get-EC2SpotPriceHistory -InstanceType c3.large -AvailabilityZone us-west-2a -
MaxResult 10
```

輸出：

```
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
...
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeSpotPriceHistory](#)。

Get-EC2Subnet

以下程式碼範例顯示如何使用 Get-EC2Subnet。

PowerShell V4 的工具

範例 1：此範例說明指定的子網路。

```
Get-EC2Subnet -SubnetId subnet-1a2b3c4d
```

輸出：

```
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
```

範例 2：此範例說明您的所有子網路。

```
Get-EC2Subnet
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeSubnets](#)。

Get-EC2Tag

以下程式碼範例顯示如何使用 Get-EC2Tag。

PowerShell V4 的工具

範例 1：此範例會擷取資源類型 'image' 的標籤

```
Get-EC2Tag -Filter @{"Name"="resource-type";Values="image"}
```

輸出：

Key	ResourceId	ResourceType	Value
---	-----	-----	----
Name	ami-0a123b4ccb567a8ea	image	Win7-Imported
auto-delete	ami-0a123b4ccb567a8ea	image	never

範例 2：此範例會擷取所有資源的所有標籤，並依資源類型分組

```
Get-EC2Tag | Group-Object resourcetype
```

輸出：

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}
```

範例 3：此範例顯示指定區域具有 'auto-delete' 標籤且值為 'no' 的所有資源

```
Get-EC2Tag -Region eu-west-1 -Filter @{"Name"="tag:auto-delete";Values="no"}
```

輸出：

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

範例 4：此範例會取得具有 'no' 值之標籤 'auto-delete' 的所有資源，並在下一個管道中進一步篩選，以僅剖析 'instance' 資源類型，並最終為每個執行個體資源建立 'ThisInstance' 標籤，其值為執行個體 ID 本身

```
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}}
```

範例 5：此範例會擷取所有執行個體資源和 'Name' 金鑰的標籤，並以資料表格式顯示

```
Get-EC2Tag -Filter @{"Name"="resource-
type";Values="instance"},@{"Name"="key";Values="Name"} | Select-Object ResourceId,
@{"Name"="Name-Tag";Expression={$PSItem.Value}} | Format-Table -AutoSize
```

輸出：

ResourceId	Name-Tag
i-012e3cb4df567e1aa	jump1
i-01c23a45d6fc7a89f	repro-3

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeTags](#)。

Get-EC2Volume

以下程式碼範例顯示如何使用 Get-EC2Volume。

PowerShell V4 的工具

範例 1：此範例說明指定的 EBS 磁碟區。

```
Get-EC2Volume -VolumeId vol-12345678
```

輸出：

```
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
```

範例 2：此範例說明狀態為「可用」的 EBS 磁碟區。

```
Get-EC2Volume -Filter @{ Name="status"; Values="available" }
```

輸出：

```
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
...
```

範例 3：此範例說明所有 EBS 磁碟區。

```
Get-EC2Volume
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeVolumes](#)。

Get-EC2VolumeAttribute

以下程式碼範例顯示如何使用 Get-EC2VolumeAttribute。

PowerShell V4 的工具

範例 1：此範例說明指定磁碟區的指定屬性。

```
Get-EC2VolumeAttribute -VolumeId vol-12345678 -Attribute AutoEnableIO
```

輸出：

AutoEnableIO	ProductCodes	VolumeId
-----	-----	-----
False	{}	vol-12345678

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeVolumeAttribute](#)。

Get-EC2VolumeStatus

以下程式碼範例顯示如何使用 Get-EC2VolumeStatus。

PowerShell V4 的工具

範例 1：此範例說明指定磁碟區的狀態。

```
Get-EC2VolumeStatus -VolumeId vol-12345678
```

輸出：

```
Actions           : {}
AvailabilityZone   : us-west-2a
Events            : {}
VolumeId          : vol-12345678
VolumeStatus      : Amazon.EC2.Model.VolumeStatusInfo
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus
```

輸出：

Details	Status
-----	-----
{io-enabled, io-performance}	ok

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus.Details
```

輸出：

Name	Status
----	-----
io-enabled	passed
io-performance	not-applicable

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeVolumeStatus](#)。

Get-EC2Vpc

以下程式碼範例顯示如何使用 Get-EC2Vpc。

PowerShell V4 的工具

範例 1：此範例說明指定的 VPC。

```
Get-EC2Vpc -VpcId vpc-12345678
```

輸出：

```
CidrBlock      : 10.0.0.0/16
DhcpOptionsId  : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault      : False
State          : available
Tags           : {Name}
VpcId          : vpc-12345678
```

範例 2：此範例說明預設 VPC（每個區域只能有一個）。如果您的帳戶在此區域支援 EC2-Classic，則沒有預設 VPC。

```
Get-EC2Vpc -Filter @{"Name"="isDefault"; Values="true"}
```

輸出：

```
CidrBlock      : 172.31.0.0/16
DhcpOptionsId  : dopt-12345678
InstanceTenancy : default
IsDefault      : True
State          : available
Tags           : {}
VpcId          : vpc-45678901
```

範例 3：此範例描述符合指定篩選條件的 VPCs（即具有符合值 '10.0.0.0/16' 且處於狀態 'available' 的 CIDR）。

```
Get-EC2Vpc -Filter @{"Name"="cidr";
  Values="10.0.0.0/16"},@{"Name"="state";Values="available"}
```

範例 4：此範例說明您的所有 VPCs。

```
Get-EC2Vpc
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeVpcs](#)。

Get-EC2VpcAttribute

以下程式碼範例顯示如何使用 Get-EC2VpcAttribute。

PowerShell V4 的工具

範例 1：此範例說明 'enableDnsSupport' 屬性。

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsSupport
```

輸出：

```
EnableDnsSupport
-----
True
```

範例 2：此範例說明 'enableDnsHostnames' 屬性。

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsHostnames
```

輸出：

```
EnableDnsHostnames
-----
True
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeVpcAttribute](#)。

Get-EC2VpcClassicLink

以下程式碼範例顯示如何使用 Get-EC2VpcClassicLink。

PowerShell V4 的工具

範例 1：上述範例會傳回具有該區域 ClassicLinkEnabled 狀態的所有 VPCs

```
Get-EC2VpcClassicLink -Region eu-west-1
```

輸出：

```
ClassicLinkEnabled Tags      VpcId
-----
-----
```

```
False          {Name} vpc-0fc1ff23f45b678eb
False          {}      vpc-01e23c4a5d6db78e9
False          {Name} vpc-0123456b078b9d01f
False          {}      vpc-12cf3b4f
False          {Name} vpc-0b12d3456a7e8901d
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeVpcClassicLink](#)。

Get-EC2VpcClassicLinkDnsSupport

以下程式碼範例顯示如何使用 Get-EC2VpcClassicLinkDnsSupport。

PowerShell V4 的工具

範例 1：此範例說明區域 eu-west-1 的 VPCs ClassicLink DNS 支援狀態

```
Get-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

輸出：

```
ClassicLinkDnsSupported VpcId
-----
False                  vpc-0b12d3456a7e8910d
False                  vpc-12cf3b4f
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeVpcClassicLinkDnsSupport](#)。

Get-EC2VpcEndpoint

以下程式碼範例顯示如何使用 Get-EC2VpcEndpoint。

PowerShell V4 的工具

範例 1：此範例說明區域 eu-west-1 的一或多個 VPC 端點。然後，它會將輸出輸送至下一個命令，選取 VpcEndpointId 屬性，並將陣列 VPC ID 傳回為字串陣列

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object -ExpandProperty VpcEndpointId
```

輸出：

```
vpce-01a2ab3f4f5cc6f7d
vpce-01d2b345a6787890b
vpce-0012e34d567890e12
vpce-0c123db4567890123
```

範例 2：此範例說明區域 eu-west-1 的所有 vpc 端點，並選取 VpcEndpointId、VpcId、ServiceName 和 PrivateDnsEnabled 屬性以表格格式呈現

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object VpcEndpointId, VpcId,
ServiceName, PrivateDnsEnabled | Format-Table -AutoSize
```

輸出：

VpcEndpointId	VpcId	ServiceName	PrivateDnsEnabled
-----	-----	-----	-----
vpce-02a2ab2f2f2cc2f2d	vpce-0fc6ff46f65b039eb	com.amazonaws.eu-west-1.ssm	True
vpce-01d1b111a1114561b	vpce-0fc6ff46f65b039eb	com.amazonaws.eu-west-1.ec2	True
vpce-0011e23d45167e838	vpce-0fc6ff46f65b039eb	com.amazonaws.eu-west-1.ec2messages	True
vpce-0c123db4567890123	vpce-0fc6ff46f65b039eb	com.amazonaws.eu-west-1.ssmmessages	True

範例 3：此範例會將 VPC 端點 vpce-01a2ab3f4f5cc6f7d 的政策文件匯出至 json 檔案

```
Get-EC2VpcEndpoint -Region eu-west-1 -VpcEndpointId vpce-01a2ab3f4f5cc6f7d | Select-
Object -expand PolicyDocument | Out-File vpce_policyDocument.json
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeVpcEndpoints](#)。

Get-EC2VpcEndpointService

以下程式碼範例顯示如何使用 Get-EC2VpcEndpointService。

PowerShell V4 的工具

範例 1：此範例描述具有指定篩選條件的 EC2 VPC 端點服務，在此情況下為 `com.amazonaws.eu-west-1.ecs`。此外，它還會擴展 `ServiceDetails` 屬性並顯示詳細資訊

```
Get-EC2VpcEndpointService -Region eu-west-1 -MaxResult 5 -Filter @{"Name"="service-name";Values="com.amazonaws.eu-west-1.ecs"} | Select-Object -ExpandProperty ServiceDetails
```

輸出：

```
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
```

範例 2：此範例會擷取所有 EC2 VPC 端點服務，並傳回符合 "ssm" 的 `ServiceNames`

```
Get-EC2VpcEndpointService -Region eu-west-1 | Select-Object -ExpandProperty Servicenames | Where-Object { -match "ssm"}
```

輸出：

```
com.amazonaws.eu-west-1.ssm
com.amazonaws.eu-west-1.ssmmessages
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeVpcEndpointServices](#)。

Get-EC2VpnConnection

以下程式碼範例顯示如何使用 `Get-EC2VpnConnection`。

PowerShell V4 的工具

範例 1：此範例說明指定的 VPN 連線。


```
Get-EC2VpnConnection -VpnConnectionId vpn-12345678
```

輸出：

```
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
```

範例 2：此範例說明狀態為待定或可用的任何 VPN 連接。

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnConnection -Filter $filter
```

範例 3：此範例說明您的所有 VPN 連線。

```
Get-EC2VpnConnection
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeVpnConnections](#)。

Get-EC2VpnGateway

以下程式碼範例顯示如何使用 Get-EC2VpnGateway。

PowerShell V4 的工具

範例 1：此範例說明指定的虛擬私有閘道。

```
Get-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

輸出：

```
AvailabilityZone :  
State           : available  
Tags            : {}  
Type            : ipsec.1  
VpcAttachments  : {vpc-12345678}  
VpnGatewayId    : vgw-1a2b3c4d
```

範例 2：此範例說明狀態為待定或可用的任何虛擬私有閘道。

```
$filter = New-Object Amazon.EC2.Model.Filter  
$filter.Name = "state"  
$filter.Values = @( "pending", "available" )  
  
Get-EC2VpnGateway -Filter $filter
```

範例 3：此範例說明所有虛擬私有閘道。

```
Get-EC2VpnGateway
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeVpnGateways](#)。

Grant-EC2SecurityGroupEgress

以下程式碼範例顯示如何使用 Grant-EC2SecurityGroupEgress。

PowerShell V4 的工具

範例 1：此範例定義 EC2-VPC 指定安全群組的輸出規則。此規則會授予 TCP 連接埠 80 上指定 IP 地址範圍的存取權。此範例使用的語法需要 PowerShell 第 3 版或更新版本。

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

範例 2：使用 PowerShell 第 2 版時，您必須使用 New-Object 來建立 IpPermission 物件。

```
$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
```

範例 3：此範例會授予 TCP 連接埠 80 上指定來源安全群組的存取權。

```
$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 } )
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [AuthorizeSecurityGroupEgress](#)。

Grant-EC2SecurityGroupIngress

以下程式碼範例顯示如何使用 Grant-EC2SecurityGroupIngress。

PowerShell V4 的工具

範例 1：此範例定義 EC2-VPC 安全群組的輸入規則。這些規則會授予 SSH (連接埠 22) 和 RDC (連接埠 3389) 特定 IP 地址的存取權。請注意，您必須使用安全群組 ID 而非安全群組名稱來識別 EC2-VPC 的安全群組。此範例使用的語法需要 PowerShell 第 3 版或更新版本。

```
$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 )
```

範例 2：使用 PowerShell 第 2 版時，您必須使用 New-Object 來建立 IpPermission 物件。

```
$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 )
```

範例 3：此範例定義 EC2-Classic 安全群組的輸入規則。這些規則會授予 SSH (連接埠 22) 和 RDC (連接埠 3389) 特定 IP 地址的存取權。此範例使用的語法需要 PowerShell 第 3 版或更新版本。

```
$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 )
```

範例 4：使用 PowerShell 第 2 版，您必須使用 New-Object 來建立 IpPermission 物件。

```
$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 )
```

範例 5：此範例會將來自指定來源安全群組 (sg-1a2b3c4d) 的 TCP 連接埠 8081 存取權授予指定的安全群組 (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 } )
```

範例 6：此範例會將 CIDR 5.5.5.5/32 新增至安全群組 sg-1234abcd 的傳入規則，以用於 TCP 連接埠 22 流量和描述。

```
$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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [AuthorizeSecurityGroupIngress](#)。

Import-EC2Image

以下程式碼範例顯示如何使用 Import-EC2Image。

PowerShell V4 的工具

範例 1：此範例會使用幕等性字符，將單一磁碟虛擬機器映像從指定的 Amazon S3 儲存貯體匯入 Amazon EC2。此範例要求具有預設名稱 'vmimport' 的 VM Import Service 角色存在，其政策允許 Amazon EC2 存取指定的儲存貯體，如 VM Import Prerequisites 主題所述。若要使用自訂角色，請使用 **-RoleName** 參數指定角色名稱。

```
$container = New-Object Amazon.EC2.Model.ImageDiskContainer
$container.Format="VMDK"
$container.UserBucket = New-Object Amazon.EC2.Model.UserBucket
$container.UserBucket.S3Bucket = "amzn-s3-demo-bucket"
$container.UserBucket.S3Key = "Win_2008_Server_Standard_SP2_64-bit-disk1.vmdk"

$params = @{
    "ClientToken"="idempotencyToken"
    "Description"="Windows 2008 Standard Image Import"
    "Platform"="Windows"
```

```
"LicenseType"="AWS"
}

Import-EC2Image -DiskContainer $container @parms
```

輸出：

```
Architecture      :
Description       : Windows 2008 Standard Image
Hypervisor        :
ImageId           :
ImportTaskId      : import-ami-abcdefgh
LicenseType       : AWS
Platform          : Windows
Progress          : 2
SnapshotDetails   : {}
Status            : active
StatusMessage     : pending
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ImportImage](#)。

Import-EC2KeyPair

以下程式碼範例顯示如何使用 Import-EC2KeyPair。

PowerShell V4 的工具

範例 1：此範例會將公有金鑰匯入 EC2。第一行將公有金鑰檔案 (*.pub) 的內容存放在變數中。**\$publickey** 接著，範例會將公有金鑰檔案的 UTF8 格式轉換為 Base64-encoded 字串，並將轉換後的字串存放在變數中 **\$pkbase64**。在最後一行中，轉換後的公有金鑰會匯入 EC2。Cmdlet 會傳回金鑰指紋和名稱做為結果。

```
$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
```

輸出：

KeyFingerprint	KeyName

```
-----
do:d0:15:8f:79:97:12:be:00:fd:df:31:z3:b1:42:z1 Example-user-key
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [ImportKeyPair](#)。

Import-EC2Snapshot

以下程式碼範例顯示如何使用 Import-EC2Snapshot。

PowerShell V4 的工具

範例 1：此範例會將格式為 'VMDK' 的 VM 磁碟映像匯入 Amazon EBS 快照。此範例需要預設名稱為 'vmimport' 的 VM Import Service 角色，以及允許 Amazon EC2 存取指定儲存貯體的政策，如 <http://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/VMImportPrerequisites.html>：// 中的 **VM Import Prerequisites** 主題所述。若要使用自訂角色，請使用 **-RoleName** 參數指定角色名稱。

```
$parms = @{
    "ClientToken"="idempotencyToken"
    "Description"="Disk Image Import"
    "DiskContainer_Description" = "Data disk"
    "DiskContainer_Format" = "VMDK"
    "DiskContainer_S3Bucket" = "amzn-s3-demo-bucket"
    "DiskContainer_S3Key" = "datadiskimage.vmdk"
}

Import-EC2Snapshot @parms
```

輸出：

Description	ImportTaskId	SnapshotTaskDetail
-----	-----	-----
Disk Image Import	import-snap-abcdefgh	
Amazon.EC2.Model.SnapshotTaskDetail		

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ImportSnapshot](#)。

Move-EC2AddressToVpc

以下程式碼範例顯示如何使用 Move-EC2AddressToVpc。

PowerShell V4 的工具

範例 1：此範例會將公有 IP 地址為 12.345.67.89 的 EC2 執行個體移至美國東部（維吉尼亞北部）區域的 EC2-VPC 平台。

```
Move-EC2AddressToVpc -PublicIp 12.345.67.89 -Region us-east-1
```

範例 2：此範例會將 Get-EC2Instance 命令的結果輸送至 Move-EC2AddressToVpc cmdlet。Get-EC2Instance 命令會取得由執行個體 ID 指定的執行個體，然後傳回執行個體的公有 IP 地址屬性。

```
(Get-EC2Instance -Instance i-12345678).Instances.PublicIpAddress | Move-EC2AddressToVpc
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [MoveAddressToVpc](#)。

New-EC2Address

以下程式碼範例顯示如何使用 New-EC2Address。

PowerShell V4 的工具

範例 1：此範例會配置要與 VPC 中的執行個體搭配使用的彈性 IP 地址。

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

輸出：

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

範例 2：此範例會配置彈性 IP 地址，以與 EC2-Classic 中的執行個體搭配使用。

```
New-EC2Address
```


輸出：

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [AllocateAddress](#)。

New-EC2CustomerGateway

以下程式碼範例顯示如何使用 New-EC2CustomerGateway。

PowerShell V4 的工具

範例 1：此範例會建立指定的客戶閘道。

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

輸出：

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateCustomerGateway](#)。

New-EC2DhcpOption

以下程式碼範例顯示如何使用 New-EC2DhcpOption。

PowerShell V4 的工具

範例 1：此範例會建立指定的 DHCP 選項集。此範例使用的語法需要 PowerShell 版本 3 或更新版本。

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

輸出：

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

範例 2：使用 PowerShell 第 2 版時，您必須使用 `New-Object` 來建立每個 DHCP 選項。

```
$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)
```

輸出：

DhcpConfigurations	DhcpOptionsId	Tags
-----	-----	----
{domain-name, domain-name-servers}	dopt-2a3b4c5d	{}

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateDhcpOptions](#)。

New-EC2FlowLog

以下程式碼範例顯示如何使用 `New-EC2FlowLog`。

PowerShell V4 的工具

範例 1：此範例會針對子網路子網路 1d234567 建立 EC2 流程日誌，並使用 'Admin' 角色的滲透，針對所有 'REJECT' 流量建立名為 'subnet1-log' 的 cloud-watch-log

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

輸出：

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [CreateFlowLogs](#)。

New-EC2Host

以下程式碼範例顯示如何使用 New-EC2Host。

PowerShell V4 的工具

範例 1：此範例會將指定執行個體類型和可用區域的專用主機配置給您的帳戶

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

輸出：

```
h-01e23f4cd567890f3
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [AllocateHosts](#)。

New-EC2HostReservation

以下程式碼範例顯示如何使用 New-EC2HostReservation。

PowerShell V4 的工具

範例 1：此範例會購買預留服務 hro-0c1f23456789d0ab，其組態與您專用主機 h-01e23f4cd567890f1 的組態相符

```
New-EC2HostReservation -OfferingId hro-0c1f23456789d0ab HostIdSet  
h-01e23f4cd567890f1
```

輸出：

```
ClientToken      :  
CurrencyCode     :  
Purchase         : {hr-0123f4b5d67bedc89}  
TotalHourlyPrice : 1.307  
TotalUpfrontPrice : 0.000
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [PurchaseHostReservation](#)。

New-EC2Image

以下程式碼範例顯示如何使用 New-EC2Image。

PowerShell V4 的工具

範例 1：此範例會從指定的執行個體建立具有指定名稱和描述的 AMI。Amazon EC2 會在建立映像之前嘗試徹底關閉執行個體，並在完成時重新啟動執行個體。

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web  
server AMI"
```

範例 2：此範例會從指定的執行個體建立具有指定名稱和描述的 AMI。Amazon EC2 在不關閉和重新啟動執行個體的情況下建立映像；因此，無法保證所建立映像上的檔案系統完整性。

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web  
server AMI" -NoReboot $true
```

範例 3：此範例會建立具有三個磁碟區的 AMI。第一個磁碟區是以 Amazon EBS 快照為基礎。第二個磁碟區是空的 100 GiB Amazon EBS 磁碟區。第三個磁碟區是執行個體存放區磁碟區。此範例使用的語法需要 PowerShell 第 3 版或更新版本。

```
$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"})
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [CreateImage](#)。

New-EC2Instance

以下程式碼範例顯示如何使用 New-EC2Instance。

PowerShell V4 的工具

範例 1：此範例會在 EC2-Classic 或預設 VPC 中啟動指定 AMI 的單一執行個體。

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -InstanceType
m3.medium -KeyName my-key-pair -SecurityGroup my-security-group
```

範例 2：此範例會在 VPC 中啟動指定 AMI 的單一執行個體。

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -SubnetId
subnet-12345678 -InstanceType t2.micro -KeyName my-key-pair -SecurityGroupId
sg-12345678
```

範例 3：若要新增 EBS 磁碟區或執行個體存放區磁碟區，請定義區塊型設備映射並將其新增至命令。此範例會新增執行個體存放區磁碟區。

```
$bdm = New-Object Amazon.EC2.Model.BlockDeviceMapping
$bdm.VirtualName = "ephemeral0"
$bdm.DeviceName = "/dev/sdf"
```

```
New-EC2Instance -ImageId ami-12345678 -BlockDeviceMapping $bdm ...
```

範例 4：若要指定其中一個目前的 Windows AMIs，請使用 Get-EC2ImageByName 取得其 AMI ID。此範例會從適用於 Windows Server 2016 的目前基本 AMI 啟動執行個體。

```
$ami = Get-EC2ImageByName WINDOWS_2016_BASE
```

```
New-EC2Instance -ImageId $ami.ImageId ...
```

範例 5：在指定的專用主機環境中啟動執行個體。

```
New-EC2Instance -ImageId ami-1a2b3c4d -InstanceType m4.large -KeyName my-key-pair
-SecurityGroupId sg-1a2b3c4d -AvailabilityZone us-west-1a -Tenancy host -HostID
h-1a2b3c4d5e6f1a2b3
```

範例 6：此請求會啟動兩個執行個體，並將具有 Web 伺服器金鑰和生產值的標籤套用至執行個體。請求也會將具有成本中心索引鍵和 cc123 值的標籤套用至建立的磁碟區（在此案例中為每個執行個體的根磁碟區）。

```
$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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [RunInstances](#)。

New-EC2InstanceExportTask

以下程式碼範例顯示如何使用 New-EC2InstanceExportTask。

PowerShell V4 的工具

範例 1：此範例會將已停止的執行個體 **i-0800b00a00EXAMPLE** 作為虛擬硬碟 (VHD) 匯出至 S3 儲存貯體 **testbucket-export-instances-2019**。目標環境為 **Microsoft**，且新增區域參數，因為執行個體位於 **us-east-1** 區域，而使用者的預設 AWS 區域不是 us-east-1。若要取得匯出任務的狀態，請從此命令的結果複製 **ExportTaskId** 值，然後執行 **Get-EC2ExportTask -ExportTaskId export_task_ID_from_results**。

```
New-EC2InstanceExportTask -InstanceId i-0800b00a00EXAMPLE -
ExportToS3Task_DiskImageFormat VHD -ExportToS3Task_S3Bucket "amzn-s3-demo-bucket" -
TargetEnvironment Microsoft -Region us-east-1
```

輸出：

```
Description      :
ExportTaskId      : export-i-077c73108aEXAMPLE
ExportToS3Task    : Amazon.EC2.Model.ExportToS3Task
InstanceExportDetails : Amazon.EC2.Model.InstanceExportDetails
State             : active
StatusMessage     :
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateInstanceExportTask](#)。

New-EC2InternetGateway

以下程式碼範例顯示如何使用 New-EC2InternetGateway。

PowerShell V4 的工具

範例 1：此範例會建立網際網路閘道。

```
New-EC2InternetGateway
```

輸出：

Attachments	InternetGatewayId	Tags
-----	-----	----
{}	igw-1a2b3c4d	{}

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateInternetGateway](#)。

New-EC2KeyPair

以下程式碼範例顯示如何使用 New-EC2KeyPair。

PowerShell V4 的工具

範例 1：此範例會建立金鑰對，並在具有指定名稱的檔案中擷取 PEM 編碼的 RSA 私有金鑰。當您使用 PowerShell 時，編碼必須設定為 `ascii` 才能產生有效的金鑰。如需詳細資訊，請參閱《AWS 命令列界面使用者指南》中的建立、顯示和刪除 Amazon EC2 金鑰對 (<https://docs.aws.amazon.com/cli/latest/userguide/cli-services-ec2-keypairs.html> : //)。

```
(New-EC2KeyPair -KeyName "my-key-pair").KeyMaterial | Out-File -Encoding ascii -
FilePath C:\path\my-key-pair.pem
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateKeyPair](#)。

New-EC2NetworkAcl

以下程式碼範例顯示如何使用 `New-EC2NetworkAcl`。

PowerShell V4 的工具

範例 1：此範例會為指定的 VPC 建立網路 ACL。

```
New-EC2NetworkAcl -VpcId vpc-12345678
```

輸出：

```
Associations : {}
Entries       : {Amazon.EC2.Model.NetworkAclEntry, Amazon.EC2.Model.NetworkAclEntry}
IsDefault     : False
NetworkAclId  : acl-12345678
Tags          : {}
VpcId         : vpc-12345678
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateNetworkAcl](#)。

New-EC2NetworkAclEntry

以下程式碼範例顯示如何使用 `New-EC2NetworkAclEntry`。

PowerShell V4 的工具

範例 1：此範例會為指定的網路 ACL 建立項目。此規則允許來自 UDP 連接埠 53 (DNS) 上任何位置 (0.0.0.0/0) 的傳入流量傳入任何相關聯的子網路。

```
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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateNetworkAclEntry](#)。

New-EC2NetworkInterface

以下程式碼範例顯示如何使用 New-EC2NetworkInterface。

PowerShell V4 的工具

範例 1：此範例會建立指定的網路介面。

```
New-EC2NetworkInterface -SubnetId subnet-1a2b3c4d -Description "my network  
interface" -Group sg-12345678 -PrivateIpAddress 10.0.0.17
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateNetworkInterface](#)。

New-EC2PlacementGroup

以下程式碼範例顯示如何使用 New-EC2PlacementGroup。

PowerShell V4 的工具

範例 1：此範例會建立具有指定名稱的置放群組。

```
New-EC2PlacementGroup -GroupName my-placement-group -Strategy cluster
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreatePlacementGroup](#)。

New-EC2Route

以下程式碼範例顯示如何使用 New-EC2Route。

PowerShell V4 的工具

範例 1：此範例會為指定的路由表建立指定的路由。路由符合所有流量，並將其傳送至指定的網際網路閘道。

```
New-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0 -GatewayId igw-1a2b3c4d
```

輸出：

```
True
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateRoute](#)。

New-EC2RouteTable

以下程式碼範例顯示如何使用 New-EC2RouteTable。

PowerShell V4 的工具

範例 1：此範例會為指定的 VPC 建立路由表。

```
New-EC2RouteTable -VpcId vpc-12345678
```

輸出：

```
Associations      : {}  
PropagatingVgws   : {}  
Routes            : {}  
RouteTableId      : rtb-1a2b3c4d  
Tags              : {}  
VpcId             : vpc-12345678
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateRouteTable](#)。

New-EC2ScheduledInstance

以下程式碼範例顯示如何使用 New-EC2ScheduledInstance。

PowerShell V4 的工具

範例 1：此範例會啟動指定的排程執行個體。

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [RunScheduledInstances](#)。

New-EC2ScheduledInstancePurchase

以下程式碼範例顯示如何使用 New-EC2ScheduledInstancePurchase。

PowerShell V4 的工具

範例 1：此範例會購買排程執行個體。

```
$request = New-Object Amazon.EC2.Model.PurchaseRequest
$request.InstanceCount = 1
$request.PurchaseToken = "eyJ2IjoiMSIsInMiOjEsImMiOi..."
New-EC2ScheduledInstancePurchase -PurchaseRequest $request
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [PurchaseScheduledInstances](#)。

New-EC2SecurityGroup

以下程式碼範例顯示如何使用 New-EC2SecurityGroup。

PowerShell V4 的工具

範例 1：此範例會為指定的 VPC 建立安全群組。

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security group" -
VpcId vpc-12345678
```

輸出：

```
sg-12345678
```

範例 2：此範例會建立 EC2-Classic 的安全群組。

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security group"
```

輸出：

```
sg-45678901
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateSecurityGroup](#)。

New-EC2Snapshot

以下程式碼範例顯示如何使用 New-EC2Snapshot。

PowerShell V4 的工具

範例 1：此範例會建立指定磁碟區的快照。

```
New-EC2Snapshot -VolumeId vol-12345678 -Description "This is a test"
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [CreateSnapshot](#)。

New-EC2SpotDatafeedSubscription

以下程式碼範例顯示如何使用 New-EC2SpotDatafeedSubscription。

PowerShell V4 的工具

範例 1：此範例會建立 Spot 執行個體資料饋送。

```
New-EC2SpotDatafeedSubscription -Bucket amzn-s3-demo-bucket -Prefix spotdata
```

輸出：

```
Bucket   : amzn-s3-demo-bucket
Fault    :
OwnerId  : 123456789012
Prefix   : spotdata
State    : Active
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateSpotDatafeedSubscription](#)。

New-EC2Subnet

以下程式碼範例顯示如何使用 New-EC2Subnet。

PowerShell V4 的工具

範例 1：此範例會建立具有指定 CIDR 的子網路。

```
New-EC2Subnet -VpcId vpc-12345678 -CidrBlock 10.0.0.0/24
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateSubnet](#)。

New-EC2Tag

以下程式碼範例顯示如何使用 New-EC2Tag。

PowerShell V4 的工具

範例 1：此範例會將單一標籤新增至指定的資源。標籤索引鍵為 'myTag'，標籤值為 'myTagValue'。此範例使用的語法需要 PowerShell 第 3 版或更新版本。

```
New-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag"; Value="myTagValue" }
```

範例 2：此範例會更新或新增指定的標籤至指定的資源。此範例使用的語法需要 PowerShell 第 3 版或更新版本。

```
New-EC2Tag -Resource i-12345678 -Tag @( @{ Key="myTag"; Value="newTagValue" },
    @{ Key="test"; Value="anotherTagValue" } )
```

範例 3：使用 PowerShell 第 2 版時，您必須使用 New-Object 來建立標籤參數的標籤。

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

New-EC2Tag -Resource i-12345678 -Tag $tag
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [CreateTags](#)。

New-EC2Volume

以下程式碼範例顯示如何使用 New-EC2Volume。

PowerShell V4 的工具

範例 1：此範例會建立指定的磁碟區。

```
New-EC2Volume -Size 50 -AvailabilityZone us-west-2a -VolumeType gp2
```

輸出：

```
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
```

範例 2：此範例請求會建立磁碟區，並套用具有堆疊索引鍵和生產值的標籤。

```
$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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateVolume](#)。

New-EC2Vpc

以下程式碼範例顯示如何使用 New-EC2Vpc。

PowerShell V4 的工具

範例 1：此範例會建立具有指定 CIDR 的 VPC。Amazon VPC 也會為 VPC 建立下列項目：預設 DHCP 選項集、主要路由表和預設網路 ACL。


```
New-EC2VPC -CidrBlock 10.0.0.0/16
```

輸出：

```
CidrBlock      : 10.0.0.0/16
DhcpOptionsId  : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault      : False
State          : pending
Tags           : {}
VpcId          : vpc-12345678
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateVpc](#)。

New-EC2VpcEndpoint

以下程式碼範例顯示如何使用 New-EC2VpcEndpoint。

PowerShell V4 的工具

範例 1：此範例會在 VPC vpc-0fc1ff23f45b678eb 中為服務 com.amazonaws.eu-west-1.s3 建立新的 VPC 端點

```
New-EC2VpcEndpoint -ServiceName com.amazonaws.eu-west-1.s3 -VpcId
vpc-0fc1ff23f45b678eb
```

輸出：

```
ClientToken VpcEndpoint
-----
Amazon.EC2.Model.VpcEndpoint
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateVpcEndpoint](#)。

New-EC2VpnConnection

以下程式碼範例顯示如何使用 New-EC2VpnConnection。

PowerShell V4 的工具

範例 1：此範例會在指定的虛擬私有閘道與指定的客戶閘道之間建立 VPN 連線。輸出包含網路管理員所需的 XML 格式組態資訊。

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
vgw-1a2b3c4d
```

輸出：

```
CustomerGatewayConfiguration : [XML document]
CustomerGatewayId            : cgw-1a2b3c4d
Options                      :
Routes                      : {}
State                       : pending
Tags                       : {}
Type                       :
VgwTelemetry                : {}
VpnConnectionId            : vpn-12345678
VpnGatewayId                : vgw-1a2b3c4d
```

範例 2：此範例會建立 VPN 連線，並在具有指定名稱的檔案中擷取組態。

```
(New-EC2VpnConnection -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
vgw-1a2b3c4d).CustomerGatewayConfiguration | Out-File C:\path\vpn-configuration.xml
```

範例 3：此範例會在指定的虛擬私有閘道與指定的客戶閘道之間建立具有靜態路由的 VPN 連線。

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
vgw-1a2b3c4d -Options_StaticRoutesOnly $true
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateVpnConnection](#)。

New-EC2VpnConnectionRoute

以下程式碼範例顯示如何使用 New-EC2VpnConnectionRoute。

PowerShell V4 的工具

範例 1：此範例會為指定的 VPN 連接建立指定的靜態路由。

```
New-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock 11.12.0.0/16
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateVpnConnectionRoute](#)。

New-EC2VpnGateway

以下程式碼範例顯示如何使用 New-EC2VpnGateway。

PowerShell V4 的工具

範例 1：此範例會建立指定的虛擬私有閘道。

```
New-EC2VpnGateway -Type ipsec.1
```

輸出：

```
AvailabilityZone :  
State           : available  
Tags            : {}  
Type            : ipsec.1  
VpcAttachments  : {}  
VpnGatewayId    : vgw-1a2b3c4d
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateVpnGateway](#)。

Register-EC2Address

以下程式碼範例顯示如何使用 Register-EC2Address。

PowerShell V4 的工具

範例 1：此範例會將指定的彈性 IP 地址與 VPC 中指定的執行個體建立關聯。

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

輸出：

```
eipassoc-12345678
```

範例 2：此範例會將指定的彈性 IP 地址與 EC2-Classical 中指定的執行個體建立關聯。

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [AssociateAddress](#)。

Register-EC2DhcpOption

以下程式碼範例顯示如何使用 Register-EC2DhcpOption。

PowerShell V4 的工具

範例 1：此範例會將指定的 DHCP 選項集與指定的 VPC 建立關聯。

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

範例 2：此範例會將預設 DHCP 選項集與指定的 VPC 建立關聯。

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [AssociateDhcpOptions](#)。

Register-EC2Image

以下程式碼範例顯示如何使用 Register-EC2Image。

PowerShell V4 的工具

範例 1：此範例使用 Amazon S3 中指定的資訊清單檔案註冊 AMI。

```
Register-EC2Image -ImageLocation amzn-s3-demo-bucket/my-web-server-ami/  
image.manifest.xml -Name my-web-server-ami
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [RegisterImage](#)。

Register-EC2PrivateIpAddress

以下程式碼範例顯示如何使用 Register-EC2PrivateIpAddress。

PowerShell V4 的工具

範例 1：此範例會將指定的次要私有 IP 地址指派給指定的網路介面。

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

範例 2：此範例會建立兩個次要私有 IP 地址，並將其指派給指定的網路介面。

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

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [AssignPrivateIpAddresses](#)。

Register-EC2RouteTable

以下程式碼範例顯示如何使用 Register-EC2RouteTable。

PowerShell V4 的工具

範例 1：此範例會將指定的路由表與指定的子網路建立關聯。

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

輸出：

```
rtbassoc-12345678
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [AssociateRouteTable](#)。

Remove-EC2Address

以下程式碼範例顯示如何使用 Remove-EC2Address。

PowerShell V4 的工具

範例 1：此範例會發行 VPC 中執行個體的指定彈性 IP 地址。

```
Remove-EC2Address -AllocationId eipalloc-12345678 -Force
```

範例 2：此範例會發行 EC2-Classic 中執行個體的指定彈性 IP 地址。

```
Remove-EC2Address -PublicIp 198.51.100.2 -Force
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ReleaseAddress](#)。

Remove-EC2CapacityReservation

以下程式碼範例顯示如何使用 Remove-EC2CapacityReservation。

PowerShell V4 的工具

範例 1：此範例會取消容量保留 cr-0c1f2345db6f7cdba

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

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CancelCapacityReservation](#)。

Remove-EC2CustomerGateway

以下程式碼範例顯示如何使用 Remove-EC2CustomerGateway。

PowerShell V4 的工具

範例 1：此範例會刪除指定的客戶閘道。除非您也指定 Force 參數，否則在操作繼續之前會提示您進行確認。

```
Remove-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

輸出：

```
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"):
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteCustomerGateway](#)。

Remove-EC2DhcpOption

以下程式碼範例顯示如何使用 Remove-EC2DhcpOption。

PowerShell V4 的工具

範例 1：此範例會刪除指定的 DHCP 選項集。除非您也指定 Force 參數，否則在操作繼續之前會提示您進行確認。

```
Remove-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d
```

輸出：

```
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"):
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteDhcpOptions](#)。

Remove-EC2FlowLog

以下程式碼範例顯示如何使用 Remove-EC2FlowLog。

PowerShell V4 的工具

範例 1：此範例會移除指定的 FlowLogId fl-01a2b3456a789c01

```
Remove-EC2FlowLog -FlowLogId fl-01a2b3456a789c01
```

輸出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2FlowLog (DeleteFlowLogs)" on target
"fl-01a2b3456a789c01".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"): Y
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteFlowLogs](#)。

Remove-EC2Host

以下程式碼範例顯示如何使用 Remove-EC2Host。

PowerShell V4 的工具

範例 1：此範例發行指定的主機 ID h-0badafd1dcb2f3456

```
Remove-EC2Host -HostId h-0badafd1dcb2f3456
```

輸出：

```
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} {}	

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ReleaseHosts](#)。

Remove-EC2Instance

以下程式碼範例顯示如何使用 Remove-EC2Instance。

PowerShell V4 的工具

範例 1：此範例會終止指定的執行個體（執行個體可能正在執行或處於「已停止」狀態）。Cmdlet 會在繼續之前提示確認；使用 -Force 切換來隱藏提示。

```
Remove-EC2Instance -InstanceId i-12345678
```

輸出：

CurrentState	InstanceId	PreviousState
-----	-----	-----
Amazon.EC2.Model.InstanceState	i-12345678	Amazon.EC2.Model.InstanceState

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [TerminateInstances](#)。

Remove-EC2InternetGateway

以下程式碼範例顯示如何使用 Remove-EC2InternetGateway。

PowerShell V4 的工具

範例 1：此範例會刪除指定的網際網路閘道。除非您也指定 Force 參數，否則在操作繼續之前會提示您進行確認。

```
Remove-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

輸出：

```
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"):
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteInternetGateway](#)。

Remove-EC2KeyPair

以下程式碼範例顯示如何使用 Remove-EC2KeyPair。

PowerShell V4 的工具

範例 1：此範例會刪除指定的金鑰對。除非您也指定 Force 參數，否則在操作繼續之前會提示您進行確認。

```
Remove-EC2KeyPair -KeyName my-key-pair
```

輸出：

```
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"):
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteKeyPair](#)。

Remove-EC2NetworkAcl

以下程式碼範例顯示如何使用 Remove-EC2NetworkAcl。

PowerShell V4 的工具

範例 1：此範例會刪除指定的網路 ACL。除非您也指定 Force 參數，否則在操作繼續之前會提示您進行確認。

```
Remove-EC2NetworkAcl -NetworkAclId acl-12345678
```

輸出：

```
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"):
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteNetworkAcl](#)。

Remove-EC2NetworkAclEntry

以下程式碼範例顯示如何使用 Remove-EC2NetworkAclEntry。

PowerShell V4 的工具

範例 1：此範例會從指定的網路 ACL 移除指定的規則。除非您也指定 Force 參數，否則在操作繼續之前會提示您進行確認。

```
Remove-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100
```

輸出：

```
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"):
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteNetworkAclEntry](#)。

Remove-EC2NetworkInterface

以下程式碼範例顯示如何使用 Remove-EC2NetworkInterface。

PowerShell V4 的工具

範例 1：此範例會刪除指定的網路介面。除非您也指定 Force 參數，否則在操作繼續之前會提示您進行確認。

```
Remove-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

輸出：

```
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"):
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteNetworkInterface](#)。

Remove-EC2PlacementGroup

以下程式碼範例顯示如何使用 Remove-EC2PlacementGroup。

PowerShell V4 的工具

範例 1：此範例會刪除指定的置放群組。除非您也指定 Force 參數，否則在操作繼續之前會提示您進行確認。

```
Remove-EC2PlacementGroup -GroupName my-placement-group
```

輸出：

```
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"):
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeletePlacementGroup](#)。

Remove-EC2Route

以下程式碼範例顯示如何使用 Remove-EC2Route。

PowerShell V4 的工具

範例 1：此範例會從指定的路由表中刪除指定的路由。除非您也指定 Force 參數，否則在操作繼續之前會提示您進行確認。

```
Remove-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0
```

輸出：

```
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"):
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteRoute](#)。

Remove-EC2RouteTable

以下程式碼範例顯示如何使用 Remove-EC2RouteTable。

PowerShell V4 的工具

範例 1：此範例會刪除指定的路由表。除非您也指定 Force 參數，否則系統會提示您在操作繼續之前進行確認。

```
Remove-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

輸出：

```
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"):
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteRouteTable](#)。

Remove-EC2SecurityGroup

以下程式碼範例顯示如何使用 Remove-EC2SecurityGroup。

PowerShell V4 的工具

範例 1：此範例會刪除 EC2-VPC 的指定安全群組。除非您也指定 Force 參數，否則在操作繼續之前會提示您進行確認。

```
Remove-EC2SecurityGroup -GroupId sg-12345678
```

輸出：

```
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"):
```

範例 2：此範例會刪除 EC2-Classic 的指定安全群組。

```
Remove-EC2SecurityGroup -GroupName my-security-group -Force
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteSecurityGroup](#)。

Remove-EC2Snapshot

以下程式碼範例顯示如何使用 Remove-EC2Snapshot。

PowerShell V4 的工具

範例 1：此範例會刪除指定的快照。除非您也指定 Force 參數，否則系統會提示您在操作繼續之前進行確認。

```
Remove-EC2Snapshot -SnapshotId snap-12345678
```

輸出：

```
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"):
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteSnapshot](#)。

Remove-EC2SpotDatafeedSubscription

以下程式碼範例顯示如何使用 Remove-EC2SpotDatafeedSubscription。

PowerShell V4 的工具

範例 1：此範例會刪除 Spot 執行個體資料饋送。除非您也指定 Force 參數，否則在操作繼續之前會提示您進行確認。

```
Remove-EC2SpotDatafeedSubscription
```

輸出：

```
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"):
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteSpotDatafeedSubscription](#)。

Remove-EC2Subnet

以下程式碼範例顯示如何使用 Remove-EC2Subnet。

PowerShell V4 的工具

範例 1：此範例會刪除指定的子網路。除非您也指定 Force 參數，否則在操作繼續之前會提示您進行確認。

```
Remove-EC2Subnet -SubnetId subnet-1a2b3c4d
```

輸出：

```
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"):
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteSubnet](#)。

Remove-EC2Tag

以下程式碼範例顯示如何使用 Remove-EC2Tag。

PowerShell V4 的工具

範例 1：此範例會從指定的資源刪除指定的標籤，無論標籤值為何。此範例使用的語法需要 PowerShell 版本 3 或更新版本。

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag" } -Force
```

範例 2：此範例會從指定的資源刪除指定的標籤，但僅限於標籤值相符時。此範例使用的語法需要 PowerShell 版本 3 或更新版本。

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag";Value="myTagValue" } -Force
```

範例 3：此範例會從指定的資源刪除指定的標籤，無論標籤值為何。

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```


範例 4：此範例會從指定的資源刪除指定的標籤，但僅限於標籤值相符時。

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteTags](#)。

Remove-EC2Volume

以下程式碼範例顯示如何使用 Remove-EC2Volume。

PowerShell V4 的工具

範例 1：此範例會分離指定的磁碟區。除非您也指定 Force 參數，否則在操作繼續之前會提示您進行確認。

```
Remove-EC2Volume -VolumeId vol-12345678
```

輸出：

```
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"):
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteVolume](#)。

Remove-EC2Vpc

以下程式碼範例顯示如何使用 Remove-EC2Vpc。

PowerShell V4 的工具

範例 1：此範例會刪除指定的 VPC。除非您也指定 Force 參數，否則在操作繼續之前會提示您進行確認。

```
Remove-EC2Vpc -VpcId vpc-12345678
```

輸出：

```
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"):
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteVpc](#)。

Remove-EC2VpnConnection

以下程式碼範例顯示如何使用 Remove-EC2VpnConnection。

PowerShell V4 的工具

範例 1：此範例會刪除指定的 VPN 連線。除非您也指定 Force 參數，否則在操作繼續之前會提示您進行確認。

```
Remove-EC2VpnConnection -VpnConnectionId vpn-12345678
```

輸出：

```
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"):
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteVpnConnection](#)。

Remove-EC2VpnConnectionRoute

以下程式碼範例顯示如何使用 Remove-EC2VpnConnectionRoute。

PowerShell V4 的工具

範例 1：此範例會從指定的 VPN 連接移除指定的靜態路由。除非您也指定 Force 參數，否則在操作繼續之前會提示您進行確認。

```
Remove-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock 11.12.0.0/16
```

輸出：

```
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"):
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteVpnConnectionRoute](#)。

Remove-EC2VpnGateway

以下程式碼範例顯示如何使用 Remove-EC2VpnGateway。

PowerShell V4 的工具

範例 1：此範例會刪除指定的虛擬私有閘道。除非您也指定 Force 參數，否則在操作繼續之前會提示您進行確認。

```
Remove-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

輸出：

```
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"):
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteVpnGateway](#)。

Request-EC2SpotFleet

以下程式碼範例顯示如何使用 Request-EC2SpotFleet。

PowerShell V4 的工具

範例 1：此範例會在可用區域中以指定執行個體類型的最低價格建立 Spot 機群請求。如果您的帳戶僅支援 EC2-VPC，Spot 機群會在具有預設子網路的最低價可用區域中啟動執行個體。如果您的帳戶支援 EC2-Classic，Spot 機群會在價格最低的可用區域中啟動 EC2-Classic 中的執行個體。請注意，您支付的價格不會超過請求的指定 Spot 價格。

```
$sg = New-Object Amazon.EC2.Model.GroupIdentifier
$sg.GroupId = "sg-12345678"
$lc = New-Object Amazon.EC2.Model.SpotFleetLaunchSpecification
$lc.ImageId = "ami-12345678"
$lc.InstanceType = "m3.medium"
$lc.SecurityGroups.Add($sg)
Request-EC2SpotFleet -SpotFleetRequestConfig_SpotPrice 0.04 `
-SpotFleetRequestConfig_TargetCapacity 2 `
-SpotFleetRequestConfig_IamFleetRole arn:aws:iam::123456789012:role/my-spot-fleet-
role `
-SpotFleetRequestConfig_LaunchSpecification $lc
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [RequestSpotFleet](#)。

Request-EC2SpotInstance

以下程式碼範例顯示如何使用 Request-EC2SpotInstance。

PowerShell V4 的工具

範例 1：此範例會請求指定子網路中的一次性 Spot 執行個體。請注意，必須為包含指定子網路的 VPC 建立安全群組，並且必須使用網路界面透過 ID 指定。當您指定網路界面時，您必須使用網路界面包含子網路 ID。

```
$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
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [RequestSpotInstances](#)。

Reset-EC2ImageAttribute

以下程式碼範例顯示如何使用 Reset-EC2ImageAttribute。

PowerShell V4 的工具

範例 1：此範例會將 'launchPermission' 屬性重設為其預設值。根據預設，AMIs 是私有的。

```
Reset-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ResetImageAttribute](#)。

Reset-EC2InstanceAttribute

以下程式碼範例顯示如何使用 Reset-EC2InstanceAttribute。

PowerShell V4 的工具

範例 1：此範例會重設指定執行個體的 'sriovNetSupport' 屬性。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

範例 2：此範例會重設指定執行個體的 'ebsOptimized' 屬性。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

範例 3：此範例會重設指定執行個體的 'sourceDestCheck' 屬性。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sourceDestCheck
```

範例 4：此範例會重設指定執行個體的 'disableApiTermination' 屬性。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

範例 5：此範例會重設指定執行個體的 'instanceInitiatedShutdownBehavior' 屬性。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute  
instanceInitiatedShutdownBehavior
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ResetInstanceAttribute](#)。

Reset-EC2NetworkInterfaceAttribute

以下程式碼範例顯示如何使用 Reset-EC2NetworkInterfaceAttribute。

PowerShell V4 的工具

範例 1：此範例會重設指定網路介面的來源/目的地檢查。

```
Reset-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -SourceDestCheck
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [ResetNetworkInterfaceAttribute](#)。

Reset-EC2SnapshotAttribute

以下程式碼範例顯示如何使用 Reset-EC2SnapshotAttribute。

PowerShell V4 的工具

範例 1：此範例會重設指定快照的指定屬性。

```
Reset-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute  
CreateVolumePermission
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [ResetSnapshotAttribute](#)。

Restart-EC2Instance

以下程式碼範例顯示如何使用 Restart-EC2Instance。

PowerShell V4 的工具

範例 1：此範例會重新啟動指定的執行個體。

```
Restart-EC2Instance -InstanceId i-12345678
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [RebootInstances](#)。

Revoke-EC2SecurityGroupEgress

以下程式碼範例顯示如何使用 Revoke-EC2SecurityGroupEgress。

PowerShell V4 的工具

範例 1：此範例會移除 EC2-VPC 指定安全群組的規則。這會撤銷 TCP 連接埠 80 上指定 IP 地址範圍的存取權。此範例使用的語法需要 PowerShell 第 3 版或更新版本。

```
$ip = @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; IpRanges="203.0.113.0/24" }
```

```
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

範例 2：使用 PowerShell 第 2 版時，您必須使用 `New-Object` 來建立 `IpPermission` 物件。

```
$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
```

範例 3：此範例會撤銷對 TCP 連接埠 80 上指定來源安全群組的存取權。

```
$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 }
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [RevokeSecurityGroupEgress](#)。

Revoke-EC2SecurityGroupIngress

以下程式碼範例顯示如何使用 `Revoke-EC2SecurityGroupIngress`。

PowerShell V4 的工具

範例 1：此範例會從 EC2-VPC 指定安全群組的指定地址範圍撤銷對 TCP 連接埠 22 的存取。請注意，您必須使用安全群組 ID 而非安全群組名稱來識別 EC2-VPC 的安全群組。此範例使用的語法需要 PowerShell 第 3 版或更新版本。

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.0/24" }
Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

範例 2：使用 PowerShell 第 2 版時，您必須使用 `New-Object` 來建立 `IpPermission` 物件。

```
$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
```

範例 3：此範例會從 EC2-Classic 指定安全群組的指定地址範圍撤銷對 TCP 連接埠 22 的存取。此範例使用的語法需要 PowerShell 第 3 版或更新版本。

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.0/24" }

Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

範例 4：使用 PowerShell 第 2 版時，您必須使用 New-Object 來建立 IpPermission 物件。

```
$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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [RevokeSecurityGroupIngress](#)。

Send-EC2InstanceStatus

以下程式碼範例顯示如何使用 Send-EC2InstanceStatus。

PowerShell V4 的工具

範例 1：此範例會報告指定執行個體的狀態意見回饋。

```
Send-EC2InstanceStatus -Instance i-12345678 -Status impaired -ReasonCode
unresponsive
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ReportInstanceStatus](#)。

Set-EC2NetworkAclAssociation

以下程式碼範例顯示如何使用 Set-EC2NetworkAclAssociation。

PowerShell V4 的工具

範例 1：此範例會將指定的網路 ACL 與指定網路 ACL 關聯的子網路建立關聯。

```
Set-EC2NetworkAclAssociation -NetworkAclId acl-12345678 -AssociationId  
aclassoc-1a2b3c4d
```

輸出：

```
aclassoc-87654321
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [ReplaceNetworkAclAssociation](#)。

Set-EC2NetworkAclEntry

以下程式碼範例顯示如何使用 Set-EC2NetworkAclEntry。

PowerShell V4 的工具

範例 1：此範例會取代指定網路 ACL 的指定項目。新規則允許從指定地址到任何相關聯子網路的傳入流量。

```
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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [ReplaceNetworkAclEntry](#)。

Set-EC2Route

以下程式碼範例顯示如何使用 Set-EC2Route。

PowerShell V4 的工具

範例 1：此範例會取代指定路由表的指定路由。新路由會將指定的流量傳送至指定的虛擬私有閘道。

```
Set-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 10.0.0.0/24 -GatewayId
vgw-1a2b3c4d
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ReplaceRoute](#)。

Set-EC2RouteTableAssociation

以下程式碼範例顯示如何使用 Set-EC2RouteTableAssociation。

PowerShell V4 的工具

範例 1：此範例會將指定的路由表與指定路由表關聯的子網路建立關聯。

```
Set-EC2RouteTableAssociation -RouteTableId rtb-1a2b3c4d -AssociationId
rtbassoc-12345678
```

輸出：

```
rtbassoc-87654321
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ReplaceRouteTableAssociation](#)。

Start-EC2Instance

以下程式碼範例顯示如何使用 Start-EC2Instance。

PowerShell V4 的工具

範例 1：此範例會啟動指定的執行個體。

```
Start-EC2Instance -InstanceId i-12345678
```

輸出：

CurrentState	InstanceId	PreviousState
-----	-----	-----
Amazon.EC2.Model.InstanceState	i-12345678	Amazon.EC2.Model.InstanceState

範例 2：此範例會啟動指定的執行個體。

```
@("i-12345678", "i-76543210") | Start-EC2Instance
```

範例 3：此範例會啟動目前停止的一組執行個體。Get-EC2Instance 傳回的執行個體物件會輸送至 Start-EC2Instance。此範例使用的語法需要 PowerShell 第 3 版或更新版本。

```
(Get-EC2Instance -Filter @{ Name="instance-state-name"; Values="stopped"}).Instances  
| Start-EC2Instance
```

範例 4：使用 PowerShell 第 2 版時，您必須使用 New-Object 來建立篩選條件參數的篩選條件。

```
$filter = New-Object Amazon.EC2.Model.Filter  
$filter.Name = "instance-state-name"  
$filter.Values = "stopped"  
  
(Get-EC2Instance -Filter $filter).Instances | Start-EC2Instance
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [StartInstances](#)。

Start-EC2InstanceMonitoring

以下程式碼範例顯示如何使用 Start-EC2InstanceMonitoring。

PowerShell V4 的工具

範例 1：此範例會啟用指定執行個體的詳細監控。

```
Start-EC2InstanceMonitoring -InstanceId i-12345678
```

輸出：

InstanceId	Monitoring
-----	-----
i-12345678	Amazon.EC2.Model.Monitoring

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [MonitorInstances](#)。

Stop-EC2ImportTask

以下程式碼範例顯示如何使用 Stop-EC2ImportTask。

PowerShell V4 的工具

範例 1：此範例會取消指定的匯入任務（快照或映像匯入）。如有必要，可以使用 -**CancelReason** 參數提供原因。

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CancelImportTask](#)。

Stop-EC2Instance

以下程式碼範例顯示如何使用 Stop-EC2Instance。

PowerShell V4 的工具

範例 1：此範例會停止指定的執行個體。

```
Stop-EC2Instance -InstanceId i-12345678
```

輸出：

CurrentState	InstanceId	PreviousState
-----	-----	-----
Amazon.EC2.Model.InstanceState	i-12345678	Amazon.EC2.Model.InstanceState

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [StopInstances](#)。

Stop-EC2InstanceMonitoring

以下程式碼範例顯示如何使用 Stop-EC2InstanceMonitoring。

PowerShell V4 的工具

範例 1：此範例會停用指定執行個體的詳細監控。

```
Stop-EC2InstanceMonitoring -InstanceId i-12345678
```

輸出：

```
InstanceId      Monitoring
-----
i-12345678      Amazon.EC2.Model.Monitoring
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [UnmonitorInstances](#)。

Stop-EC2SpotFleetRequest

以下程式碼範例顯示如何使用 Stop-EC2SpotFleetRequest。

PowerShell V4 的工具

範例 1：此範例會取消指定的 Spot 機群請求，並終止相關聯的 Spot 執行個體。

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

範例 2：此範例會取消指定的 Spot 機群請求，而不終止相關聯的 Spot 執行個體。

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [CancelSpotFleetRequests](#)。

Stop-EC2SpotInstanceRequest

以下程式碼範例顯示如何使用 Stop-EC2SpotInstanceRequest。

PowerShell V4 的工具

範例 1：此範例會取消指定的 Spot 執行個體請求。

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

輸出：

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [CancelSpotInstanceRequests](#)。

Unregister-EC2Address

以下程式碼範例顯示如何使用 Unregister-EC2Address。

PowerShell V4 的工具

範例 1：此範例會將指定的彈性 IP 地址與 VPC 中指定的執行個體取消關聯。

```
Unregister-EC2Address -AssociationId eipassoc-12345678
```

範例 2：此範例會取消指定彈性 IP 地址與 EC2-Classical 中指定執行個體的關聯。

```
Unregister-EC2Address -PublicIp 203.0.113.17
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DisassociateAddress](#)。

Unregister-EC2Image

以下程式碼範例顯示如何使用 Unregister-EC2Image。

PowerShell V4 的工具

範例 1：此範例會取消註冊指定的 AMI。

```
Unregister-EC2Image -ImageId ami-12345678
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeregisterImage](#)。

Unregister-EC2PrivateIpAddress

以下程式碼範例顯示如何使用 Unregister-EC2PrivateIpAddress。

PowerShell V4 的工具

範例 1：此範例會從指定的網路界面取消指派指定的私有 IP 地址。

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UnassignPrivateIpAddresses](#)。

Unregister-EC2RouteTable

以下程式碼範例顯示如何使用 Unregister-EC2RouteTable。

PowerShell V4 的工具

範例 1：此範例會移除路由表與子網路之間的指定關聯。

```
Unregister-EC2RouteTable -AssociationId rtbassoc-1a2b3c4d
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DisassociateRouteTable](#)。

Update-EC2SecurityGroupRuleIngressDescription

以下程式碼範例顯示如何使用 Update-EC2SecurityGroupRuleIngressDescription。

PowerShell V4 的工具

範例 1：更新現有傳入（傳入）安全群組規則的描述。

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupId  
"sgr-1234567890"  
$ruleWithUpdatedDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{  
    "SecurityGroupId" = $existingInboundRule.SecurityGroupId  
    "Description" = "Updated rule description"
```



```
}  
  
Update-EC2SecurityGroupRuleIngressDescription -GroupId $existingInboundRule.GroupId  
-SecurityGroupRuleDescription $ruleWithUpdatedDescription
```

範例 2：移除現有傳入（傳入）安全群組規則的描述（透過省略請求中的參數）。

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupRuleId  
"sgr-1234567890"  
$ruleWithoutDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{  
    "SecurityGroupRuleId" = $existingInboundRule.SecurityGroupRuleId  
}  
  
Update-EC2SecurityGroupRuleIngressDescription -GroupId $existingInboundRule.GroupId  
-SecurityGroupRuleDescription $ruleWithoutDescription
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [UpdateSecurityGroupRuleDescriptionsIngress](#)。

使用 Tools for PowerShell V4 的 Amazon ECR 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 Amazon ECR 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Get-ECRLoginCommand

以下程式碼範例顯示如何使用 Get-ECRLoginCommand。

PowerShell V4 的工具

範例 1：傳回 PObject，其中包含登入資訊，可用於向 IAM 主體可存取的任何 Amazon ECR 登錄進行身分驗證。呼叫取得授權字串所需的登入資料和區域端點是從 shell 預設值（由 **Set-AWSCredential/Set-DefaultAWSRegion** 或 **Initialize-AWSDefaultConfiguration** cmdlet 設定）取得。您可以使用 Command 屬性搭配 Invoke-Expression 來登入指定的登錄檔，或在需要登入的其他工具中使用傳回的登入資料。

```
Get-ECRLoginCommand
```

輸出：

```
Username       : AWS
Password       : eyJwYXlsb2Fk...kRBVEffS0VZIn0=
ProxyEndpoint  : https://123456789012.dkr.ecr.us-west-2.amazonaws.com
Endpoint       : https://123456789012.dkr.ecr.us-west-2.amazonaws.com
ExpiresAt      : 9/26/2017 6:08:23 AM
Command        : docker login --username AWS --password
eyJwYXlsb2Fk...kRBVEffS0VZIn0= https://123456789012.dkr.ecr.us-west-2.amazonaws.com
```

範例 2：擷取 PObject，其中包含您用來做為 docker 登入命令輸入的登入資訊。只要 IAM 主體可以存取該登錄檔，您就可以指定要驗證的任何 Amazon ECR 登錄 URI。

```
(Get-ECRLoginCommand).Password | docker login --username AWS --password-stdin
012345678910.dkr.ecr.us-east-1.amazonaws.com
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [Get-ECRLoginCommand](#)。

使用 Tools for PowerShell V4 的 Amazon ECS 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 Amazon ECS 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執程式碼的指示。

主題

- [動作](#)

動作

Get-ECSClusterDetail

以下程式碼範例顯示如何使用 Get-ECSClusterDetail。

PowerShell V4 的工具

範例 1：此 cmdlet 說明一或多個 ECS 叢集。

```
Get-ECSClusterDetail -Cluster "LAB-ECS-CL" -Include SETTINGS | Select-Object *
```

輸出：

```
LoggedAt          : 12/27/2019 9:27:41 PM
Clusters          : {LAB-ECS-CL}
Failures          : {}
ResponseMetadata  : Amazon.Runtime.ResponseMetadata
ContentLength     : 396
HttpStatusCode    : OK
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeClusters](#)。

Get-ECSClusterList

以下程式碼範例顯示如何使用 Get-ECSClusterList。

PowerShell V4 的工具

範例 1：此 cmdlet 會傳回現有 ECS 叢集的清單。

```
Get-ECSClusterList
```

輸出：

```
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS-CL
```

```
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListClusters](#)。

Get-ECSClusterService

以下程式碼範例顯示如何使用 Get-ECSClusterService。

PowerShell V4 的工具

範例 1：此範例列出預設叢集中執行的所有服務。

```
Get-ECSClusterService
```

範例 2：此範例列出在指定叢集中執行的所有服務。

```
Get-ECSClusterService -Cluster myCluster
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListServices](#)。

Get-ECSService

以下程式碼範例顯示如何使用 Get-ECSService。

PowerShell V4 的工具

範例 1：此範例示範如何從預設叢集擷取特定服務的詳細資訊。

```
Get-ECSService -Service my-http-service
```

範例 2：此範例示範如何擷取在具名叢集中執行之特定服務的詳細資訊。

```
Get-ECSService -Cluster myCluster -Service my-http-service
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeServices](#)。

New-ECSCluster

以下程式碼範例顯示如何使用 New-ECSCluster。

PowerShell V4 的工具

範例 1：此 cmdlet 會建立新的 Amazon ECS 叢集。

```
New-ECSCluster -ClusterName "LAB-ECS-CL" -Setting @{"Name="containerInsights";  
Value="enabled"}
```

輸出：

```
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                    : {}
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateCluster](#)。

New-ECSService

以下程式碼範例顯示如何使用 New-ECSService。

PowerShell V4 的工具

範例 1：此範例命令會在您的預設叢集中建立名為 `ecs-simple-service` 的服務。服務使用 `ecs-demo` 任務定義，並維護該任務的 10 個執行個體。

```
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10
```

範例 2：此範例命令會在您預設叢集中名為 `ecs-simple-service` 的負載平衡器後方建立服務。服務使用 `ecs-demo` 任務定義，並維護該任務的 10 個執行個體。

```
$lb = @{
    LoadBalancerName = "EC2Contai-EcsElast-S06278JGSJCM"
    ContainerName = "simple-demo"
    ContainerPort = 80
}
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10 -LoadBalancer $lb
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateService](#)。

Remove-ECSCluster

以下程式碼範例顯示如何使用 Remove-ECSCluster。

PowerShell V4 的工具

範例 1：此 cmdlet 會刪除指定的 ECS 叢集。您必須從此叢集取消註冊所有容器執行個體，才能將其刪除。

```
Remove-ECSCluster -Cluster "LAB-ECS"
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteCluster](#)。

Remove-ECSService

以下程式碼範例顯示如何使用 Remove-ECSService。

PowerShell V4 的工具

範例 1：刪除預設叢集中名為 'my-http-service' 的服務。服務必須具有所需的計數並執行 0 計數，您才能將其刪除。在命令繼續之前，系統會提示您進行確認。若要略過確認提示，請新增 -Force 切換。

```
Remove-ECSService -Service my-http-service
```

範例 2：刪除具名叢集中名為 'my-http-service' 的服務。

```
Remove-ECSService -Cluster myCluster -Service my-http-service
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteService](#)。

Update-ECSClusterSetting

以下程式碼範例顯示如何使用 Update-ECSClusterSetting。

PowerShell V4 的工具

範例 1：此 cmdlet 修改用於 ECS 叢集的設定。

```
Update-ECSClusterSetting -Cluster "LAB-ECS-CL" -Setting @{Name="containerInsights";  
Value="disabled"}
```

輸出：

```
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                               : {}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [UpdateClusterSettings](#)。

Update-ECSService

以下程式碼範例顯示如何使用 Update-ECSService。

PowerShell V4 的工具

範例 1：此範例命令會更新 `my-http-service` 服務，以使用 `amazon-ecs-sample` 任務定義。

```
Update-ECSService -Service my-http-service -TaskDefinition amazon-ecs-sample
```

範例 2：此範例命令會將所需的 `my-http-service` 服務計數更新為 10。

```
Update-ECSService -Service my-http-service -DesiredCount 10
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UpdateService](#)。

使用 Tools for PowerShell V4 的 Amazon EFS 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 Amazon EFS 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執程式碼的指示。

主題

- [動作](#)

動作

Edit-EFSMountTargetSecurityGroup

以下程式碼範例顯示如何使用 Edit-EFSMountTargetSecurityGroup。

PowerShell V4 的工具

範例 1：更新指定掛載目標的有效安全群組。最多可以指定 5 個，格式為 "sg-xxxxxxx"。

```
Edit-EFSMountTargetSecurityGroup -MountTargetId fsmt-1a2b3c4d -SecurityGroup sg-group1,sg-group3
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ModifyMountTargetSecurityGroups](#)。

Get-EFSFileSystem

以下程式碼範例顯示如何使用 Get-EFSFileSystem。

PowerShell V4 的工具

範例 1：傳回 區域中發起人帳戶擁有的所有檔案系統的集合。

```
Get-EFSFileSystem
```

輸出：

```
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
...
```

範例 2：傳回指定檔案系統的詳細資訊。

```
Get-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

範例 3：使用建立檔案系統時指定的冪等性建立字串，傳回檔案系統的詳細資訊。

```
Get-EFSFileSystem -CreationToken 1a2bff54-85e0-4747-bd95-7bc172c4f555
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeFileSystems](#)。

Get-EFSMountTarget

以下程式碼範例顯示如何使用 Get-EFSMountTarget。

PowerShell V4 的工具

範例 1：傳回與指定檔案系統相關聯的掛載目標集合。

```
Get-EFSMountTarget -FileSystemId fs-1a2b3c4d
```

輸出：

```
FileSystemId      : fs-1a2b3c4d
IpAddress         : 10.0.0.131
LifecycleState    : available
MountTargetId     : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId          : 123456789012
SubnetId         : subnet-1a2b3c4d
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeMountTargets](#)。

Get-EFSMountTargetSecurityGroup

以下程式碼範例顯示如何使用 Get-EFSMountTargetSecurityGroup。

PowerShell V4 的工具

範例 1：傳回目前指派給與掛載目標相關聯之網路介面的安全群組 ID。

```
Get-EFSMountTargetSecurityGroup -MountTargetId fsmt-1a2b3c4d
```

輸出：

```
sg-1a2b3c4d
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeMountTargetSecurityGroups](#)。

Get-EFSTag

以下程式碼範例顯示如何使用 Get-EFSTag。

PowerShell V4 的工具

範例 1：傳回目前與指定檔案系統相關聯的標籤集合。

```
Get-EFSTag -FileSystemId fs-1a2b3c4d
```

輸出：

Key	Value
---	-----
Name	My File System
tagkey1	tagvalue1
tagkey2	tagvalue2

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeTags](#)。

New-EFSFileSystem

以下程式碼範例顯示如何使用 New-EFSFileSystem。

PowerShell V4 的工具

範例 1：建立新的空白檔案系統。用於確保等冪建立的字符會自動產生，並且可以從傳回物件 **CreationToken** 的成員存取。

```
New-EFSFileSystem
```

輸出：

```
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
```

範例 2：使用自訂字串建立新的空白檔案系統，以確保等冚建立。

```
New-EFSFileSystem -CreationToken "MyUniqueToken"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateFileSystem](#)。

New-EFSMountTarget

以下程式碼範例顯示如何使用 New-EFSMountTarget。

PowerShell V4 的工具

範例 1：為檔案系統建立新的掛載目標。將使用指定的子網路，來決定要建立掛載目標的 Virtual Private Cloud (VPC)，以及要自動指派的 IP 地址（從子網路的地址範圍）。指派的 IP 地址可用來在 Amazon EC2 執行個體上掛載此檔案系統。由於未指定安全群組，為目標建立的網路介面會與子網路 VPC 的預設安全群組相關聯。

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d
```

輸出：

```
FileSystemId      : fs-1a2b3c4d
IpAddress         : 10.0.0.131
LifeCycleState    : creating
MountTargetId     : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
```

```
OwnerId          : 123456789012
SubnetId         : subnet-1a2b3c4d
```

範例 2：為具有自動指派 IP 地址的指定檔案系統建立新的掛載目標。為掛載目標建立的網路界面與指定的安全群組相關聯（最多可指定 5 個，格式為 "sg-xxxxxxx"，可指定）。

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -
SecurityGroup sg-group1,sg-group2,sg-group3
```

範例 3：為具有指定 IP 地址的指定檔案系統建立新的掛載目標。

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -IpAddress
10.0.0.131
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateMountTarget](#)。

New-EFS_Tag

以下程式碼範例顯示如何使用 New-EFS_Tag。

PowerShell V4 的工具

範例 1：將標籤集合套用至指定的檔案系統。如果檔案系統上已存在具有指定金鑰的標籤，則會更新該標籤的值。

```
New-EFS_Tag -FileSystemId fs-1a2b3c4d -Tag
@{Key="tagkey1";Value="tagvalue1"},@{Key="tagkey2";Value="tagvalue2"}
```

範例 2：設定指定檔案系統的名稱標籤。使用 Get-EFSFileSystem cmdlet 時，此值會與其他檔案系統詳細資訊一起傳回。

```
New-EFS_Tag -FileSystemId fs-1a2b3c4d -Tag @{Key="Name";Value="My File System"}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateTags](#)。

Remove-EFSFileSystem

以下程式碼範例顯示如何使用 Remove-EFSFileSystem。

PowerShell V4 的工具

範例 1：刪除不再使用的指定檔案系統（如果檔案系統具有掛載目標，必須先將其移除）。在 cmdlet 繼續進行之前，系統會提示您進行確認 - 若要隱藏確認，請使用 **-Force** 開關。

```
Remove-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteFileSystem](#)。

Remove-EFSMountTarget

以下程式碼範例顯示如何使用 Remove-EFSMountTarget。

PowerShell V4 的工具

範例 1：刪除指定的掛載目標。在操作繼續之前，系統會提示您進行確認。若要隱藏提示，請使用 **-Force** 開關。請注意，此操作會透過目標強制中斷檔案系統的任何掛載，如果可行，建議您考慮在執行此命令之前卸載檔案系統。

```
Remove-EFSMountTarget -MountTargetId fsmt-1a2b3c4d
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteMountTarget](#)。

Remove-EFSTag

以下程式碼範例顯示如何使用 Remove-EFSTag。

PowerShell V4 的工具

範例 1：從檔案系統刪除一或多個標籤的集合。在 cmdlet 繼續進行之前，系統會提示您進行確認 - 若要隱藏確認，請使用 **-Force** 切換。

```
Remove-EFSTag -FileSystemId fs-1a2b3c4d -TagKey "tagkey1","tagkey2"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteTags](#)。

使用 Tools for PowerShell V4 的 Amazon EKS 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 Amazon EKS 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執程式碼的指示。

主題

- [動作](#)

動作

Add-EKSResourceTag

以下程式碼範例顯示如何使用 Add-EKSResourceTag。

PowerShell V4 的工具

範例 1：此 cmdlet 會將指定的標籤與指定的 resourceArn 建立關聯。

```
Add-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD" -  
Tag @{Name = "EKSPRODCLUSTER"}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [TagResource](#)。

Get-EKSCluster

以下程式碼範例顯示如何使用 Get-EKSCluster。

PowerShell V4 的工具

範例 1：此 cmdlet 會傳回 Amazon EKS 叢集的描述性資訊。

```
Get-EKSCluster -Name "PROD"
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeCluster](#)。

Get-EKSClusterList

以下程式碼範例顯示如何使用 Get-EKSClusterList。

PowerShell V4 的工具

範例 1：此 cmdlet 列出指定區域中您 AWS 帳戶 中的 Amazon EKS 叢集。

```
Get-EKSClusterList
```

輸出：

```
PROD
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListClusters](#)。

Get-EKSFargateProfile

以下程式碼範例顯示如何使用 Get-EKSFargateProfile。

PowerShell V4 的工具

範例 1：此 cmdlet 會傳回 AWS Fargate 描述檔的描述性資訊。

```
Get-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

輸出：

```
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             : {}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeFargateProfile](#)。

Get-EKSFargateProfileList

以下程式碼範例顯示如何使用 Get-EKSFargateProfileList。

PowerShell V4 的工具

範例 1：此 cmdlet 列出與指定 AWS 帳戶 區域中 中指定叢集相關聯的 AWS Fargate 設定檔。

```
Get-EKSFargateProfileList -ClusterName "TEST"
```

輸出：

```
EKSFargate
EKSFargateProfile
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListFargateProfiles](#)。

Get-EKSNodegroup

以下程式碼範例顯示如何使用 Get-EKSNodegroup。

PowerShell V4 的工具

範例 1：此 cmdlet 會傳回 Amazon EKS 節點群組的描述性資訊。

```
Get-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeNodegroup](#)。

Get-EKSNodegroupList

以下程式碼範例顯示如何使用 Get-EKSNodegroupList。

PowerShell V4 的工具

範例 1：此 cmdlet 列出與指定 AWS 帳戶 區域中 中指定叢集相關聯的 Amazon EKS 節點群組。

```
Get-EKSNodegroupList -ClusterName PROD
```

輸出：

```
ProdEKSNodeGroup
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListNodegroups](#)。

Get-EKSResourceTag

以下程式碼範例顯示如何使用 Get-EKSResourceTag。

PowerShell V4 的工具

範例 1：此 cmdlet 列出 Amazon EKS 資源的標籤。

```
Get-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
```

輸出：

```
Key Value
---
Name EKSPRODCLUSTER
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListTagsForResource](#)。

Get-EKSUpdate

以下程式碼範例顯示如何使用 Get-EKSUpdate。

PowerShell V4 的工具

範例 1：此 cmdlet 會針對您的 Amazon EKS 叢集或關聯的受管節點群組傳回有關更新的描述性資訊。

```
Get-EKSUpdate -Name "PROD" -UpdateId "ee708232-7d2e-4ed7-9270-d0b5176f0726"
```

輸出：

```
CreatedAt : 12/25/2019 5:03:07 PM
Errors    : {}
Id        : ee708232-7d2e-4ed7-9270-d0b5176f0726
Params    : {Amazon.EKS.Model.UpdateParam}
Status    : Successful
Type      : LoggingUpdate
```

- 如需 API 詳細資訊，請參閱《Cmdlet 參考》中的 [DescribeUpdate](#) (V4)。AWS Tools for PowerShell

Get-EKSUpdateList

以下程式碼範例顯示如何使用 Get-EKSUpdateList。

PowerShell V4 的工具

範例 1：此 cmdlet 列出 AWS 帳戶與指定區域中 Amazon EKS 叢集或受管節點群組相關聯的更新。

```
Get-EKSUpdateList -Name "PROD"
```

輸出：

```
ee708232-7d2e-4ed7-9270-d0b5176f0726
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListUpdates](#)。

New-EKSCluster

以下程式碼範例顯示如何使用 New-EKSCluster。

PowerShell V4 的工具

範例 1：此範例會建立一個名為 'prod' 的新叢集。

```
New-EKSCluster -Name prod -ResourcesVpcConfig
@{SubnetIds=@("subnet-0a1b2c3d","subnet-3a2b1c0d");SecurityGroupIds="sg-6979fe18"}
-RoleArn "arn:aws:iam::012345678901:role/eks-service-role"
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateCluster](#)。

New-EKSFargateProfile

以下程式碼範例顯示如何使用 New-EKSFargateProfile。

PowerShell V4 的工具

範例 1：此 cmdlet 會為您的 Amazon EKS 叢集建立 AWS Fargate 設定檔。叢集中必須至少有一個 Fargate 設定檔，才能在 Fargate 基礎設施上排程 Pod。

```
New-EKSFargateProfile -FargateProfileName EKSFargateProfile -ClusterName TEST -
Subnet "subnet-02f6ff500ff2067a0", "subnet-0cd976f08d5fbfaae" -PodExecutionRoleArn
arn:aws:iam::012345678912:role/AmazonEKSFargatePodExecutionRole -Selector
@{Namespace="default"}
```

輸出：

```
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               : {}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateFargateProfile](#)。

New-EKSNodeGroup

以下程式碼範例顯示如何使用 New-EKSNodeGroup。

PowerShell V4 的工具

範例 1：此 cmdlet 會為 Amazon EKS 叢集建立受管工作者節點群組。您只能為叢集建立一個等於叢集目前 Kubernetes 版本的節點群組。所有節點群組都會使用叢集個別次要 Kubernetes 版本的最新 AMI 發行版本建立。

```
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"
```

輸出：

```
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

```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateNodegroup](#)。

Remove-EKSCluster

以下程式碼範例顯示如何使用 Remove-EKSCluster。

PowerShell V4 的工具

範例 1：此 cmdlet 會刪除 Amazon EKS 叢集控制平面。

```
Remove-EKSCluster -Name "DEV-KUBE-CL"
```

輸出：

```

Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSCluster (DeleteCluster)" on target "DEV-KUBE-CL".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "Y"): Y

Arn                : arn:aws:eks:us-west-2:012345678912:cluster/DEV-KUBE-CL
CertificateAuthority : Amazon.EKS.Model.Certificate
ClientRequestToken  :
CreatedAt           : 12/25/2019 9:33:25 AM
Endpoint            : https://02E6D31E3E4F8C15D7BE7F58D527776A.yl4.us-west-2.eks.amazonaws.com
Identity            : Amazon.EKS.Model.Identity
Logging             : Amazon.EKS.Model.Logging
Name                : DEV-KUBE-CL
PlatformVersion     : eks.7
ResourcesVpcConfig  : Amazon.EKS.Model.VpcConfigResponse
RoleArn             : arn:aws:iam::012345678912:role/eks-iam-role

```

```
Status      : DELETING
Tags        : {}
Version     : 1.14
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteCluster](#)。

Remove-EKSFargateProfile

以下程式碼範例顯示如何使用 Remove-EKSFargateProfile。

PowerShell V4 的工具

範例 1：此 cmdlet 會刪除 AWS Fargate 設定檔。當您刪除 Fargate 設定檔時，任何在 Fargate 上執行且使用設定檔建立的 Pod 都會遭到刪除。

```
Remove-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

輸出：

```
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             : {}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteFargateProfile](#)。

Remove-EKSNodegroup

以下程式碼範例顯示如何使用 Remove-EKSNodegroup。

PowerShell V4 的工具

範例 1：此 cmdlet 會刪除叢集的 Amazon EKS 節點群組。

```
Remove-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteNodegroup](#)。

Remove-EKSResourceTag

以下程式碼範例顯示如何使用 Remove-EKSResourceTag。

PowerShell V4 的工具

範例 1：此 cmdlet 會從 EKS 資源刪除指定的標籤。

```
Remove-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
-TagKey "Name"
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [UntagResource](#)。

Update-EKSClusterConfig

以下程式碼範例顯示如何使用 Update-EKSClusterConfig。

PowerShell V4 的工具

範例 1：更新 Amazon EKS 叢集組態。您的叢集會在更新期間繼續運作。

```
Update-EKSClusterConfig -Name "PROD" -Logging_ClusterLogging
@{Types="api","audit","authenticator","controllerManager","scheduler",Enabled="True"}
```

輸出：

```
CreatedAt : 12/25/2019 5:03:07 PM
Errors    : {}
Id        : ee708232-7d2e-4ed7-9270-d0b5176f0726
Params    : {Amazon.EKS.Model.UpdateParam}
Status    : InProgress
```

Type : LoggingUpdate

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [UpdateClusterConfig](#)。

Update-EKSClusterVersion

以下程式碼範例顯示如何使用 Update-EKSClusterVersion。

PowerShell V4 的工具

範例 1：此 cmdlet 會將 Amazon EKS 叢集更新為指定的 Kubernetes 版本。您的叢集會在更新期間繼續運作。

```
Update-EKSClusterVersion -Name "PROD-KUBE-CL" -Version 1.14
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UpdateClusterVersion](#)。

Elastic Load Balancing - 使用 Tools for PowerShell V4 的第 1 版範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 Elastic Load Balancing - 第 1 版來執行動作並實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Add-ELBLoadBalancerToSubnet

以下程式碼範例顯示如何使用 Add-ELBLoadBalancerToSubnet。

PowerShell V4 的工具

範例 1：此範例會將指定的子網路新增至為指定負載平衡器設定的子網路集。輸出包含子網路的完整清單。

```
Add-ELBLoadBalancerToSubnet -LoadBalancerName my-load-balancer -Subnet  
subnet-12345678
```

輸出：

```
subnet-12345678  
subnet-87654321
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [AttachLoadBalancerToSubnets](#)。

Add-ELBResourceTag

以下程式碼範例顯示如何使用 Add-ELBResourceTag。

PowerShell V4 的工具

範例 1：此範例會將指定的標籤新增至指定的負載平衡器。此範例使用的語法需要 PowerShell 版本 3 或更新版本。

```
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag  
@{ Key="project";Value="lima" },@{ Key="department";Value="digital-media" }
```

範例 2：使用 PowerShell 第 2 版時，您必須使用 New-Object 為標籤參數建立標籤。

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.Tag
```

```
$tag.Key = "project"
$tag.Value = "lima"
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag $tag
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [AddTags](#)。

Disable-ELBAvailabilityZoneForLoadBalancer

以下程式碼範例顯示如何使用 Disable-ELBAvailabilityZoneForLoadBalancer。

PowerShell V4 的工具

範例 1：此範例會從指定的負載平衡器移除指定的可用區域。輸出包含剩餘的可用區域。

```
Disable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -
AvailabilityZone us-west-2a
```

輸出：

```
us-west-2b
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DisableAvailabilityZonesForLoadBalancer](#)。

Dismount-ELBLoadBalancerFromSubnet

以下程式碼範例顯示如何使用 Dismount-ELBLoadBalancerFromSubnet。

PowerShell V4 的工具

範例 1：此範例會從為指定負載平衡器設定的子網路集移除指定的子網路。輸出包含剩餘的子網路。

```
Dismount-ELBLoadBalancerFromSubnet -LoadBalancerName my-load-balancer -Subnet
subnet-12345678
```

輸出：

```
subnet-87654321
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DetachLoadBalancerFromSubnets](#)。

Edit-ELBLoadBalancerAttribute

以下程式碼範例顯示如何使用 Edit-ELBLoadBalancerAttribute。

PowerShell V4 的工具

範例 1：此範例會啟用指定負載平衡器的跨區域負載平衡。

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -  
CrossZoneLoadBalancing_Enabled $true
```

範例 2：此範例會停用指定負載平衡器的連線耗盡。

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -  
ConnectionDraining_Enabled $false
```

範例 3：此範例會啟用指定負載平衡器的存取記錄。

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer `  
>> -AccessLog_Enabled $true `  
>> -AccessLog_S3BucketName amzn-s3-demo-logging-bucket `  
>> -AccessLog_S3BucketPrefix my-app/prod `  
>> -AccessLog_EmitInterval 60
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ModifyLoadBalancerAttributes](#)。

Enable-ELBAvailabilityZoneForLoadBalancer

以下程式碼範例顯示如何使用 Enable-ELBAvailabilityZoneForLoadBalancer。

PowerShell V4 的工具

範例 1：此範例會將指定的可用區域新增至指定的負載平衡器。輸出包含可用區域的完整清單。

```
Enable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -  
AvailabilityZone us-west-2a
```

輸出：

```
us-west-2a
us-west-2b
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [EnableAvailabilityZonesForLoadBalancer](#)。

Get-ELBInstanceHealth

以下程式碼範例顯示如何使用 Get-ELBInstanceHealth。

PowerShell V4 的工具

範例 1：此範例說明向指定負載平衡器註冊的執行個體狀態。

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer
```

輸出：

Description State	InstanceId	ReasonCode
-----	-----	-----
N/A	i-87654321	N/A
InService		
Instance has failed at lea... i-12345678		Instance
OutOfService		

範例 2：此範例說明向指定負載平衡器註冊的指定執行個體狀態。

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance i-12345678
```

範例 3：此範例顯示指定執行個體狀態的完整描述。

```
(Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance  
i-12345678).Description
```

輸出：

Instance has failed at least the UnhealthyThreshold number of health checks consecutively.

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeInstanceHealth](#)。

Get-ELBLoadBalancer

以下程式碼範例顯示如何使用 Get-ELBLoadBalancer。

PowerShell V4 的工具

範例 1：此範例列出負載平衡器的名稱。

```
Get-ELBLoadBalancer | format-table -property LoadBalancerName
```

輸出：

```
LoadBalancerName
-----
my-load-balancer
my-other-load-balancer
my-internal-load-balancer
```

範例 2：此範例說明指定的負載平衡器。

```
Get-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

輸出：

```
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
```

範例 3：此範例說明目前 AWS 區域中的所有負載平衡器。

```
Get-ELBLoadBalancer
```

範例 4：此範例說明所有可用負載平衡器的所有負載平衡器 AWS 區域。

```
Get-AWSRegion | % { Get-ELBLoadBalancer -Region $_ }
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeLoadBalancers](#)。

Get-ELBLoadBalancerAttribute

以下程式碼範例顯示如何使用 Get-ELBLoadBalancerAttribute。

PowerShell V4 的工具

範例 1：此範例說明指定負載平衡器的屬性。

```
Get-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer
```

輸出：

```
AccessLog              : Amazon.ElasticLoadBalancing.Model.AccessLog
AdditionalAttributes    : {}
ConnectionDraining     : Amazon.ElasticLoadBalancing.Model.ConnectionDraining
ConnectionSettings     : Amazon.ElasticLoadBalancing.Model.ConnectionSettings
CrossZoneLoadBalancing : Amazon.ElasticLoadBalancing.Model.CrossZoneLoadBalancing
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeLoadBalancerAttributes](#)。

Get-ELBLoadBalancerPolicy

以下程式碼範例顯示如何使用 Get-ELBLoadBalancerPolicy。

PowerShell V4 的工具

範例 1：此範例說明與指定負載平衡器相關聯的政策。

```
Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer
```

輸出：

PolicyAttributeDescriptions PolicyTypeName	PolicyName
-----	-----
{ProxyProtocol}	my-ProxyProtocol-policy
ProxyProtocolPolicyType	
{CookieName}	my-app-cookie-policy
AppCookieStickinessPolicyType	

範例 2：此範例說明指定政策的屬性。

```
(Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-ProxyProtocol-policy).PolicyAttributeDescriptions
```

輸出：

AttributeName	AttributeValue
-----	-----
ProxyProtocol	true

範例 3：此範例說明預先定義的政策，包括範例政策。範例政策的名稱具有 ELBSample- 字首。

```
Get-ELBLoadBalancerPolicy
```

輸出：

PolicyAttributeDescriptions PolicyTypeName	PolicyName
-----	-----

```
{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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeLoadBalancerPolicies](#)。

Get-ELBLoadBalancerPolicyType

以下程式碼範例顯示如何使用 Get-ELBLoadBalancerPolicyType。

PowerShell V4 的工具

範例 1：此範例會取得 Elastic Load Balancing 支援的政策類型。

```
Get-ELBLoadBalancerPolicyType
```

輸出：

Description PolicyTypeName	PolicyAttributeTypeDescriptions
----- -----	-----
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... {PublicKey}
  PublicKeyPolicyType
Stickiness policy with session lifet... {CookieName}
  AppCookieStickinessPolicyType
Policy that controls whether to incl... {ProxyProtocol}
  ProxyProtocolPolicyType
```

範例 2：此範例說明指定的政策類型。

```
Get-ELBLoadBalancerPolicyType -PolicyTypeName ProxyProtocolPolicyType
```

輸出：

Description	PolicyAttributeTypeDescriptions
PolicyTypeName	
-----	-----

Policy that controls whether to incl... {ProxyProtocol}	
ProxyProtocolPolicyType	

範例 3：此範例顯示指定政策類型的完整描述。

```
(Get-ELBLoadBalancerPolicyType -PolicyTypeName).Description
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeLoadBalancerPolicyTypes](#)。

Get-ELBResourceTag

以下程式碼範例顯示如何使用 Get-ELBResourceTag。

PowerShell V4 的工具

範例 1：此範例列出指定負載平衡器的標籤。

```
Get-ELBResourceTag -LoadBalancerName @("my-load-balancer","my-internal-load-balancer")
```

輸出：

LoadBalancerName	Tags
-----	----
my-load-balancer	{project, department}
my-internal-load-balancer	{project, department}

範例 2：此範例說明指定負載平衡器的標籤。

```
(Get-ELBResourceTag -LoadBalancerName my-load-balancer).Tags
```

輸出：

Key	Value
---	-----
project	lima
department	digital-media

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeTags](#)。

Join-ELBSecurityGroupToLoadBalancer

以下程式碼範例顯示如何使用 Join-ELBSecurityGroupToLoadBalancer。

PowerShell V4 的工具

範例 1：此範例會將指定負載平衡器的目前安全群組取代為指定的安全群組。

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -
SecurityGroup sg-87654321
```

輸出：

```
sg-87654321
```

範例 2：若要保留目前的安全群組並指定額外的安全群組，請同時指定現有和新的安全群組。

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -  
SecurityGroup @("sg-12345678", "sg-87654321")
```

輸出：

```
sg-12345678  
sg-87654321
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ApplySecurityGroupsToLoadBalancer](#)。

New-ELBAppCookieStickinessPolicy

以下程式碼範例顯示如何使用 New-ELBAppCookieStickinessPolicy。

PowerShell V4 的工具

範例 1：此範例會建立黏性政策，遵循指定應用程式產生之 Cookie 的黏性工作階段生命週期。

```
New-ELBAppCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-  
app-cookie-policy -CookieName my-app-cookie
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [CreateAppCookieStickinessPolicy](#)。

New-ELBLBCookieStickinessPolicy

以下程式碼範例顯示如何使用 New-ELBLBCookieStickinessPolicy。

PowerShell V4 的工具

範例 1：此範例會建立黏性政策，其黏性工作階段生命週期由指定的過期期間（以秒為單位）控制。

```
New-ELBLBCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-  
duration-cookie-policy -CookieExpirationPeriod 60
```

範例 2：此範例會建立粘性政策，其粘性工作階段生命週期由瀏覽器的生命週期（使用者代理程式）控制。

```
New-ELBLCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-duration-cookie-policy
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [CreateLbCookieStickinessPolicy](#)。

New-ELBLoadBalancer

以下程式碼範例顯示如何使用 New-ELBLoadBalancer。

PowerShell V4 的工具

範例 1：此範例會在 VPC 中建立具有 HTTP 接聽程式的負載平衡器。

```
$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
```

範例 2：此範例會在 EC2-Classic 中建立具有 HTTP 接聽程式的負載平衡器。

```
New-ELBLoadBalancer -LoadBalancerName my-classic-load-balancer -AvailabilityZone us-west-2a -Listener $httpListener
```

輸出：

```
my-classic-load-balancer-123456789.us-west-2.elb.amazonaws.com
```

範例 3：此範例會使用 HTTPS 接聽程式建立負載平衡器。

```
$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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateLoadBalancer](#)。

New-ELBLoadBalancerListener

以下程式碼範例顯示如何使用 New-ELBLoadBalancerListener。

PowerShell V4 的工具

範例 1：此範例會將 HTTPS 接聽程式新增至指定的負載平衡器。

```
$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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [CreateLoadBalancerListeners](#)。

New-ELBLoadBalancerPolicy

以下程式碼範例顯示如何使用 New-ELBLoadBalancerPolicy。

PowerShell V4 的工具

範例 1：此範例會為指定的負載平衡器建立新的代理通訊協定政策。

```
$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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [CreateLoadBalancerPolicy](#)。

Register-ELBInstanceWithLoadBalancer

以下程式碼範例顯示如何使用 Register-ELBInstanceWithLoadBalancer。

PowerShell V4 的工具

範例 1：此範例會向指定的負載平衡器註冊指定的 EC2 執行個體。

```
Register-ELBInstanceWithLoadBalancer -LoadBalancerName my-load-balancer -Instance  
i-12345678
```

輸出：

```
InstanceId  
-----  
i-12345678  
i-87654321
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [RegisterInstancesWithLoadBalancer](#)。

Remove-ELBInstanceFromLoadBalancer

以下程式碼範例顯示如何使用 Remove-ELBInstanceFromLoadBalancer。

PowerShell V4 的工具

範例 1：此範例會從指定的負載平衡器移除指定的 EC2 執行個體。除非您也指定 Force 參數，否則在操作繼續之前會提示您進行確認。

```
Remove-ELBInstanceFromLoadBalancer -LoadBalancerName my-load-balancer -Instance  
i-12345678
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeregisterInstancesFromLoadBalancer](#)。

Remove-ELBLoadBalancer

以下程式碼範例顯示如何使用 Remove-ELBLoadBalancer。

PowerShell V4 的工具

範例 1：此範例會刪除指定的負載平衡器。除非您也指定 Force 參數，否則在操作繼續之前會提示您進行確認。

```
Remove-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

輸出：

```
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"):
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteLoadBalancer](#)。

Remove-ELBLoadBalancerListener

以下程式碼範例顯示如何使用 Remove-ELBLoadBalancerListener。

PowerShell V4 的工具

範例 1：此範例會刪除指定負載平衡器連接埠 80 上的接聽程式。除非您也指定 Force 參數，否則在操作繼續之前會提示您進行確認。

```
Remove-ELBLoadBalancerListener -LoadBalancerName my-load-balancer -LoadBalancerPort 80
```

輸出：

```
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"):
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteLoadBalancerListeners](#)。

Remove-ELBLoadBalancerPolicy

以下程式碼範例顯示如何使用 Remove-ELBLoadBalancerPolicy。

PowerShell V4 的工具

範例 1：此範例會從指定的負載平衡器刪除指定的政策。除非您也指定 Force 參數，否則在操作繼續之前會提示您進行確認。

```
Remove-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
duration-cookie-policy
```

輸出：

```
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"):
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteLoadBalancerPolicy](#)。

Remove-ELBResourceTag

以下程式碼範例顯示如何使用 Remove-ELBResourceTag。

PowerShell V4 的工具

範例 1：此範例會從指定的負載平衡器移除指定的標籤。除非您也指定 Force 參數，否則在操作繼續之前會提示您進行確認。此範例使用的語法需要 PowerShell 版本 3 或更新版本。

```
Remove-ELBResourceTag -LoadBalancerName my-load-balancer -Tag @{ Key="project" }
```

輸出：

```
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"):
```

範例 2：使用 Powershell 第 2 版時，您必須使用 New-Object 來建立標籤參數的標籤。

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.TagKeyOnly
$tag.Key = "project"
Remove-ELBResourceTag -Tag $tag -Force
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [RemoveTags](#)。

Set-ELBHealthCheck

以下程式碼範例顯示如何使用 Set-ELBHealthCheck。

PowerShell V4 的工具

範例 1：此範例會設定指定負載平衡器的運作狀態檢查設定。

```
Set-ELBHealthCheck -LoadBalancerName my-load-balancer `
>> -HealthCheck_HealthyThreshold 2 `
>> -HealthCheck_UnhealthyThreshold 2 `
>> -HealthCheck_Target "HTTP:80/ping" `
>> -HealthCheck_Interval 30 `
>> -HealthCheck_Timeout 3
```

輸出：

```
HealthyThreshold      : 2
Interval              : 30
Target                : HTTP:80/ping
Timeout               : 3
UnhealthyThreshold    : 2
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [ConfigureHealthCheck](#)。

Set-ELBLoadBalancerListenerSSLCertificate

以下程式碼範例顯示如何使用 Set-ELBLoadBalancerListenerSSLCertificate。

PowerShell V4 的工具

範例 1：此範例會取代終止指定接聽程式 SSL 連線的憑證。

```
Set-ELBLoadBalancerListenerSSLCertificate -LoadBalancerName my-load-balancer `
>> -LoadBalancerPort 443 `
>> -SSLCertificateId "arn:aws:iam::123456789012:server-certificate/new-server-cert"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [SetLoadBalancerListenerSslCertificate](#)。

Set-ELBLoadBalancerPolicyForBackendServer

以下程式碼範例顯示如何使用 Set-ELBLoadBalancerPolicyForBackendServer。

PowerShell V4 的工具

範例 1：此範例會將指定連接埠的政策取代為指定的政策。

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -  
InstancePort 80 -PolicyName my-ProxyProtocol-policy
```

範例 2：此範例會移除與指定連接埠相關聯的所有政策。

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -  
InstancePort 80
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [SetLoadBalancerPoliciesForBackendServer](#)。

Set-ELBLoadBalancerPolicyOfListener

以下程式碼範例顯示如何使用 Set-ELBLoadBalancerPolicyOfListener。

PowerShell V4 的工具

範例 1：此範例會將指定接聽程式的政策取代為指定的政策。

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -  
LoadBalancerPort 443 -PolicyName my-SSLNegotiation-policy
```

範例 2：此範例會移除與指定接聽程式相關聯的所有政策。

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -  
LoadBalancerPort 443
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [SetLoadBalancerPoliciesOfListener](#)。

Elastic Load Balancing - 使用 Tools for PowerShell V4 的第 2 版範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 Elastic Load Balancing - 第 2 版來執行動作並實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Add-ELB2ListenerCertificate

以下程式碼範例顯示如何使用 Add-ELB2ListenerCertificate。

PowerShell V4 的工具

範例 1：此範例會將其他憑證新增至指定的接聽程式。

```
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'}
```

輸出：

```
CertificateArn
IsDefault
-----
-----
arn:aws:acm:us-east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97
False
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [AddListenerCertificates](#)。

Add-ELB2Tag

以下程式碼範例顯示如何使用 Add-ELB2Tag。

PowerShell V4 的工具

範例 1：此範例會將新標籤新增至指定的 **AWS.Tools.ElasticLoadBalancingV2** 資源。

```
Add-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Tag @{Key = 'productVersion'; Value = '1.0.0'}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [AddTags](#)。

Edit-ELB2Listener

以下程式碼範例顯示如何使用 Edit-ELB2Listener。

PowerShell V4 的工具

範例 1：此範例會將指定的接聽程式預設動作修改為固定回應。

```
$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
```

輸出：

```
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        :
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ModifyListener](#)。

Edit-ELB2LoadBalancerAttribute

以下程式碼範例顯示如何使用 Edit-ELB2LoadBalancerAttribute。

PowerShell V4 的工具

範例 1：此範例會修改指定負載平衡器的屬性。

```
Edit-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Attribute @{Key =
'deletion_protection.enabled'; Value = 'true'}
```

輸出：

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ModifyLoadBalancerAttributes](#)。

Edit-ELB2Rule

以下程式碼範例顯示如何使用 Edit-ELB2Rule。

PowerShell V4 的工具

範例 1：此範例會修改指定的接聽程式規則組態。

```
$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
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ModifyRule](#)。

Edit-ELB2TargetGroup

以下程式碼範例顯示如何使用 Edit-ELB2TargetGroup。

PowerShell V4 的工具

範例 1：此範例會修改指定目標群組的屬性。

```
Edit-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -HealthCheckIntervalSecond 60 -HealthCheckPath '/index.html' -HealthCheckPort 8080
```

輸出：

```
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

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ModifyTargetGroup](#)。

Edit-ELB2TargetGroupAttribute

以下程式碼範例顯示如何使用 Edit-ELB2TargetGroupAttribute。

PowerShell V4 的工具

範例 1：此範例會修改指定目標群組的 deregistration_delay 屬性。

```

Edit-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Attribute @{Key =
'deregistration_delay.timeout_seconds'; Value = 600}

```

輸出：

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ModifyTargetGroupAttributes](#)。

Get-ELB2AccountLimit

以下程式碼範例顯示如何使用 Get-ELB2AccountLimit。

PowerShell V4 的工具

範例 1：此命令列出指定區域的 ELB2 帳戶限制。

```
Get-ELB2AccountLimit
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeAccountLimits](#)。

Get-ELB2Listener

以下程式碼範例顯示如何使用 Get-ELB2Listener。

PowerShell V4 的工具

範例 1：此範例說明指定 ALB/NLB 的接聽程式。

```
Get-ELB2Listener -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeListeners](#)。

Get-ELB2ListenerCertificate

以下程式碼範例顯示如何使用 Get-ELB2ListenerCertificate。

PowerShell V4 的工具

範例 1：此範例說明指定接聽程式的憑證。

```
Get-ELB2ListenerCertificate -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

輸出：

```
CertificateArn
IsDefault
-----
-----
```

```
arn:aws:acm:us-east-1:123456789012:certificate/5fc7c092-68bf-4862-969c-22fd48b6e17c
True
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeListenerCertificates](#)。

Get-ELB2LoadBalancer

以下程式碼範例顯示如何使用 Get-ELB2LoadBalancer。

PowerShell V4 的工具

範例 1：此範例顯示指定區域的所有負載平衡器。

```
Get-ELB2LoadBalancer
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeLoadBalancers](#)。

Get-ELB2LoadBalancerAttribute

以下程式碼範例顯示如何使用 Get-ELB2LoadBalancerAttribute。

PowerShell V4 的工具

範例 1：此命令說明指定負載平衡器的屬性。

```
Get-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/net/test-elb/238d34ad8d94bc2e'
```

輸出：

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	

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeLoadBalancerAttributes](#)。

Get-ELB2Rule

以下程式碼範例顯示如何使用 Get-ELB2Rule。

PowerShell V4 的工具

範例 1：此範例說明指定接聽程式 ARN 的接聽程式規則。

```
Get-ELB2Rule -ListenerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

輸出：

```
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

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeRules](#)。

Get-ELB2SSLPolicy

以下程式碼範例顯示如何使用 Get-ELB2SSLPolicy。

PowerShell V4 的工具

範例 1：此範例列出 ElasticLoadBalancingV2 的所有可用接聽程式政策。

```
Get-ELB2SSLPolicy
```

輸出：

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}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeSslPolicies](#)。

Get-ELB2Tag

以下程式碼範例顯示如何使用 Get-ELB2Tag。

PowerShell V4 的工具

範例 1：此範例列出指定資源的標籤。

```
Get-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

輸出：

```
ResourceArn
-----
Tags
-----
arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f {stage, internalName, version}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeTags](#)。

Get-ELB2TargetGroup

以下程式碼範例顯示如何使用 Get-ELB2TargetGroup。

PowerShell V4 的工具

範例 1：此範例說明指定的目標群組。

```
Get-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeTargetGroups](#)。

Get-ELB2TargetGroupAttribute

以下程式碼範例顯示如何使用 Get-ELB2TargetGroupAttribute。

PowerShell V4 的工具

範例 1：此範例說明指定目標群組的屬性。

```
Get-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

輸出：

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

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeTargetGroupAttributes](#)。

Get-ELB2TargetHealth

以下程式碼範例顯示如何使用 Get-ELB2TargetHealth。

PowerShell V4 的工具

範例 1：此範例會傳回指定目標群組中存在之目標的運作狀態。

```
Get-ELB2TargetHealth -TargetGroupArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

輸出：

HealthCheckPort	Target	TargetHealth
-----	-----	-----
80	Amazon.ElasticLoadBalancingV2.Model.TargetDescription	
	Amazon.ElasticLoadBalancingV2.Model.TargetHealth	

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeTargetHealth](#)。

New-ELB2Listener

以下程式碼範例顯示如何使用 New-ELB2Listener。

PowerShell V4 的工具

範例 1：此範例會使用預設動作 'Forward' 建立新的 ALB 接聽程式，將流量傳送至指定的目標群組。

```
$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
```

輸出：

```
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        :
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateListener](#)。

New-ELB2LoadBalancer

以下程式碼範例顯示如何使用 New-ELB2LoadBalancer。

PowerShell V4 的工具

範例 1：此範例使用兩個子網路建立新的面向網際網路的應用程式負載平衡器。

```
New-ELB2LoadBalancer -Type application -Scheme internet-facing -IpAddressType
  ipv4 -Name 'New-Test-ALB' -SecurityGroup 'sg-07c3414abb8811cbd' -subnet 'subnet-
  c37a67a6', 'subnet-fc02eea0'
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateLoadBalancer](#)。

New-ELB2Rule

以下程式碼範例顯示如何使用 New-ELB2Rule。

PowerShell V4 的工具

範例 1：此範例會根據指定接聽程式的客戶標頭值，使用固定回應動作建立新的接聽程式規則。

```
$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

```

輸出：

```

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

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateRule](#)。

New-ELB2TargetGroup

以下程式碼範例顯示如何使用 New-ELB2TargetGroup。

PowerShell V4 的工具

範例 1：此範例會使用提供的參數建立新的目標群組。

```

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'

```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateTargetGroup](#)。

Register-ELB2Target

以下程式碼範例顯示如何使用 Register-ELB2Target。

PowerShell V4 的工具

範例 1：此範例向指定的目標群組註冊 'i-0672a4c4cdeae3111' 執行個體。

```
Register-ELB2Target -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Target @{Port = 80; Id =
'i-0672a4c4cdeae3111'}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [RegisterTargets](#)。

Remove-ELB2Listener

以下程式碼範例顯示如何使用 Remove-ELB2Listener。

PowerShell V4 的工具

範例 1：此範例會刪除指定的接聽程式。

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

輸出：

```
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
```

範例 2：此範例會從負載平衡器移除指定的接聽程式。

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618'
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteListener](#)。

Remove-ELB2ListenerCertificate

以下程式碼範例顯示如何使用 Remove-ELB2ListenerCertificate。

PowerShell V4 的工具

範例 1：此範例會從指定的目標群組移除指定的憑證。


```
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'
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [RemoveListenerCertificates](#)。

Remove-ELB2LoadBalancer

以下程式碼範例顯示如何使用 Remove-ELB2LoadBalancer。

PowerShell V4 的工具

範例 1：此範例會刪除指定的負載平衡器。

```
Remove-ELB2LoadBalancer -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteLoadBalancer](#)。

Remove-ELB2Rule

以下程式碼範例顯示如何使用 Remove-ELB2Rule。

PowerShell V4 的工具

範例 1：此範例會從接聽程式移除指定的規則

```
Remove-ELB2Rule -RuleArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-alb/3651b4394dd9a24f/3873f123b98f7618/4b25eb10a42e33ab'
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteRule](#)。

Remove-ELB2Tag

以下程式碼範例顯示如何使用 Remove-ELB2Tag。

PowerShell V4 的工具

範例 1：此範例會移除指定金鑰的標籤。

```
Remove-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -TagKey 'productVersion'
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [RemoveTags](#)。

Remove-ELB2TargetGroup

以下程式碼範例顯示如何使用 Remove-ELB2TargetGroup。

PowerShell V4 的工具

範例 1：此範例會移除指定的目標群組。

```
Remove-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/testsssss/4e0b6076bc6483a7'
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteTargetGroup](#)。

Set-ELB2IpAddressType

以下程式碼範例顯示如何使用 Set-ELB2IpAddressType。

PowerShell V4 的工具

範例 1：此範例會將負載平衡器 IP 地址類型從 'IPv4' 變更為 'DualStack'。

```
Set-ELB2IpAddressType -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -IpAddressType dualstack
```

輸出：

```
Value
-----
dualstack
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [SetIpAddressType](#)。

Set-ELB2RulePriority

以下程式碼範例顯示如何使用 Set-ELB2RulePriority。

PowerShell V4 的工具

範例 1：此範例會變更指定接聽程式規則的優先順序。

```
Set-ELB2RulePriority -RulePriority -RulePriority @{Priority = 11; RuleArn = 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-alb/3651b4394dd9a24f/a4eb199fa5046f80/dbf4c6dcef3ec6f8'}
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [SetRulePriorities](#)。

Set-ELB2SecurityGroup

以下程式碼範例顯示如何使用 Set-ELB2SecurityGroup。

PowerShell V4 的工具

範例 1：此範例會將安全群組 'sg-07c3414abb8811cbd' 新增至指定的負載平衡器。

```
Set-ELB2SecurityGroup -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -SecurityGroup 'sg-07c3414abb8811cbd'
```

輸出：

```
sg-07c3414abb8811cbd
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [SetSecurityGroups](#)。

Set-ELB2Subnet

以下程式碼範例顯示如何使用 Set-ELB2Subnet。

PowerShell V4 的工具

範例 1：此範例會修改指定負載平衡器的子網路。

```
Set-ELB2Subnet -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Subnet 'subnet-7d8a0a51', 'subnet-c37a67a6'
```

輸出：

LoadBalancerAddresses	SubnetId	ZoneName
-----	-----	-----
{}	subnet-7d8a0a51	us-east-1c
{}	subnet-c37a67a6	us-east-1b

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [SetSubnets](#)。

Unregister-ELB2Target

以下程式碼範例顯示如何使用 Unregister-ELB2Target。

PowerShell V4 的工具

範例 1：此範例會從指定的目標群組取消註冊執行個體 'i-0672a4c4cdeae3111'。

```
$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'
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeregisterTargets](#)。

使用 Tools for PowerShell V4 的 Amazon FSx 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 Amazon FSx 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Add-FSXResourceTag

以下程式碼範例顯示如何使用 Add-FSXResourceTag。

PowerShell V4 的工具

範例 1：此範例會將標籤新增至指定的資源。

```
Add-FSXResourceTag -ResourceARN "arn:aws:fsx:eu-west-1:123456789012:file-system/  
fs-01cd23bc4bdf5678a" -Tag @{Key="Users";Value="Test"} -PassThru
```

輸出：

```
arn:aws:fsx:eu-west-1:123456789012:file-system/fs-01cd23bc4bdf5678a
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [TagResource](#)。

Get-FSXBackup

以下程式碼範例顯示如何使用 Get-FSXBackup。

PowerShell V4 的工具

範例 1：此範例會擷取自昨天為指定檔案系統 ID 建立的備份。

```
Get-FSXBackup -Filter @{Name="file-system-id";Values=$fsx.FileSystemId} | Where-Object CreationTime -gt (Get-Date).AddDays(-1)
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeBackups](#)。

Get-FSXFileSystem

以下程式碼範例顯示如何使用 Get-FSXFileSystem。

PowerShell V4 的工具

範例 1：此範例會傳回指定 `filesystemId` 的描述。

```
Get-FSXFileSystem -FileSystemId fs-01cd23bc4bdf5678a
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeFileSystems](#)。

Get-FSXResourceTagList

以下程式碼範例顯示如何使用 `Get-FSXResourceTagList`。

PowerShell V4 的工具

範例 1：此範例列出所提供資源 `arn` 的標籤。

```
Get-FSXResourceTagList -ResourceARN $fsx.ResourceARN
```

輸出：

Key	Value
---	-----
FSx-Service	Windows
Users	Dev

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListTagsForResource](#)。

New-FSXBackup

以下程式碼範例顯示如何使用 New-FSXBackup。

PowerShell V4 的工具

範例 1：此範例會建立指定檔案系統的備份。

```
New-FSXBackup -FileSystemId fs-0b1fac2345623456ba
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱《Cmdlet 參考》中的 [CreateBackup](#) (V4)。AWS Tools for PowerShell

New-FSXFileSystem

以下程式碼範例顯示如何使用 New-FSXFileSystem。

PowerShell V4 的工具

範例 1：此範例會建立新的 300GB Windows 檔案系統，允許從指定的子網路存取，支援每秒高達 8 MB 的輸送量。新的檔案系統會自動加入指定的 Microsoft Active Directory。

```
New-FSXFileSystem -FileSystemType WINDOWS -StorageCapacity
300 -SubnetId subnet-1a2b3c4d5e6f -WindowsConfiguration
@{ThroughputCapacity=8;ActiveDirectoryId='d-1a2b3c4d'}
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateFileSystem](#)。

New-FSXFileSystemFromBackup

以下程式碼範例顯示如何使用 New-FSXFileSystemFromBackup。

PowerShell V4 的工具

範例 1：此範例會從現有的 Amazon FSx for Windows File Server 備份建立新的 Amazon FSx 檔案系統。

```
New-FSXFileSystemFromBackup -BackupId $backupID -Tag @{Key="tag:Name";Value="from-
manual-backup"} -SubnetId $SubnetID -SecurityGroupId $SG_ID -WindowsConfiguration
@{ThroughputCapacity=8;ActiveDirectoryId=$DirectoryID}
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [CreateFileSystemFromBackup](#)。

Remove-FSXBackup

以下程式碼範例顯示如何使用 Remove-FSXBackup。

PowerShell V4 的工具

範例 1：此範例會移除指定的 backup-id。

```
Remove-FSXBackup -BackupId $backupID
```

輸出：

```
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

```

- 如需 API 詳細資訊，請參閱《Cmdlet 參考》中的 [DeleteBackup](#) (V4)。AWS Tools for PowerShell

Remove-FSXFileSystem

以下程式碼範例顯示如何使用 Remove-FSXFileSystem。

PowerShell V4 的工具

範例 1：此範例會移除指定的 FSX 檔案系統 ID。

```
Remove-FSXFileSystem -FileSystemId fs-012ff34e567890120
```

輸出：

```

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

```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteFileSystem](#)。

Remove-FSXResourceTag

以下程式碼範例顯示如何使用 Remove-FSXResourceTag。

PowerShell V4 的工具

範例 1：此範例會移除指定 FSX 檔案系統資源 ARN 的資源標籤。

```
Remove-FSXResourceTag -ResourceARN $FSX.ResourceARN -TagKey Users
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [UntagResource](#)。

Update-FSXFileSystem

以下程式碼範例顯示如何使用 Update-FSXFileSystem。

PowerShell V4 的工具

範例 1：此範例會透過 UpdateFileSystemWindowsConfiguration 更新 FSX 檔案系統自動備份保留天數。

```
$UpdateFSXWinConfig = [Amazon.FSx.Model.UpdateFileSystemWindowsConfiguration]::new()
$UpdateFSXWinConfig.AutomaticBackupRetentionDays = 35
Update-FSXFileSystem -FileSystemId $FSX.FileSystemId -WindowsConfiguration
$UpdateFSXWinConfig
```

輸出：

```
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

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [UpdateFileSystem](#)。

AWS Glue 使用 Tools for PowerShell V4 的範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 來執行動作和實作常見案例 AWS Glue。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

New-GLUEJob

以下程式碼範例顯示如何使用 New-GLUEJob。

PowerShell V4 的工具

範例 1：此範例會在 Glue AWS 中建立新的任務。命令名稱值一律為 **glueetl**。AWS Glue，支援執行以 Python 或 Scala 撰寫的任務指令碼。在此範例中，任務指令碼 (MyTestGlueJob.py) 是以

Python 撰寫。Python 參數是在 **\$DefArgs** 變數中指定，然後在 參數中傳遞至 PowerShell 命令，該 **DefaultArguments** 命令接受雜湊。**\$JobParams** 變數中的參數來自 CreateJob API，記錄在 Glue API 參考的任務 (<https://docs.aws.amazon.com/glue/latest/dg/aws-glue-api-jobs-job.html> : //) AWS 主題中。

```
$Command = New-Object Amazon.Glue.Model.JobCommand
$Command.Name = 'glueetl'
$Command.ScriptLocation = 's3://amzn-s3-demo-source-bucket/admin/MyTestGlueJob.py'
$Command

$Source = "source_test_table"
$Target = "target_test_table"
$Connections = $Source, $Target

$DefArgs = @{
    '--TempDir' = 's3://amzn-s3-demo-bucket/admin'
    '--job-bookmark-option' = 'job-bookmark-disable'
    '--job-language' = 'python'
}
$DefArgs

$ExecutionProp = New-Object Amazon.Glue.Model.ExecutionProperty
$ExecutionProp.MaxConcurrentRuns = 1
$ExecutionProp

$JobParams = @{
    "AllocatedCapacity"    = "5"
    "Command"              = $Command
    "Connections_Connection" = $Connections
    "DefaultArguments"     = $DefArgs
    "Description"          = "This is a test"
    "ExecutionProperty"    = $ExecutionProp
    "MaxRetries"           = "1"
    "Name"                 = "MyOregonTestGlueJob"
    "Role"                 = "Amazon-GlueServiceRoleForSSM"
    "Timeout"              = "20"
}

New-GlueJob @JobParams
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateJob](#)。

AWS Health 使用 Tools for PowerShell V4 的範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 來執行動作和實作常見案例 AWS Health。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Get-HLTHEvent

以下程式碼範例顯示如何使用 Get-HLTHEvent。

PowerShell V4 的工具

範例 1：此命令會從 AWS 個人運作狀態儀表板傳回事件。使用者新增 -Region 參數，以查看美國東部（維吉尼亞北部）區域中服務可用的事件，但 -Filter_Region 參數篩選條件適用於記錄於歐洲（倫敦）和美國西部（奧勒岡）區域 (eu-west-2 和 us-west-2) 的事件。-Filter_StartTime 參數會篩選事件可以開始的一系列時間，而 -Filter_EndTime 參數則會篩選事件可以結束的一系列時間。結果是 RDS 的排程維護事件，會在指定的 -Filter_StartTime 範圍內開始，並在排程的 -Filter_EndTime 範圍內結束。

```
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"}
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeEvents](#)。

使用 Tools for PowerShell V4 的 IAM 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 IAM 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執程式碼的指示。

主題

- [動作](#)

動作

Add-IAMClientIDToOpenIDConnectProvider

以下程式碼範例顯示如何使用 Add-IAMClientIDToOpenIDConnectProvider。

PowerShell V4 的工具

範例 1：此命令會將用戶端 ID (或對象) **my-application-ID** 新增至名為 **server.example.com** 的現有 OIDC 提供者。

```
Add-IAMClientIDToOpenIDConnectProvider -ClientID "my-application-ID"
-OpenIDConnectProviderARN "arn:aws:iam::123456789012:oidc-provider/
server.example.com"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [AddClientIDToOpenIdConnectProvider](#)。

Add-IAMRoleTag

以下程式碼範例顯示如何使用 Add-IAMRoleTag。

PowerShell V4 的工具

範例 1：此範例會標籤新增至 Identity Management Service 中的角色

```
Add-IAMRoleTag -RoleName AdminRoleaccess -Tag @{ Key = 'abac'; Value = 'testing'}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [TagRole](#)。

Add-IAMRoleToInstanceProfile

以下程式碼範例顯示如何使用 Add-IAMRoleToInstanceProfile。

PowerShell V4 的工具

範例 1：此命令會將名為 **S3Access** 的角色新增至名為 **webserver** 的現有執行個體設定檔。若要建立執行個體設定檔，請使用 **New-IAMInstanceProfile** 命令。使用此命令建立執行個體設定檔並將其與角色關聯起來之後，可以將其連接至 EC2 執行個體。為此，請使用 **New-EC2Instance** cmdlet 搭配 **InstanceProfile_Arn** 或 **InstanceProfile-Name** 參數啟動新的執行個體。

```
Add-IAMRoleToInstanceProfile -RoleName "S3Access" -InstanceProfileName "webserver"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [AddRoleToInstanceProfile](#)。

Add-IAMUserTag

以下程式碼範例顯示如何使用 Add-IAMUserTag。

PowerShell V4 的工具

範例 1：此範例會標籤新增至 Identity Management Service 中的使用者

```
Add-IAMUserTag -UserName joe -Tag @{ Key = 'abac'; Value = 'testing'}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [TagUser](#)。

Add-IAMUserToGroup

以下程式碼範例顯示如何使用 Add-IAMUserToGroup。

PowerShell V4 的工具

範例 1：此命令會將名為 **Bob** 的使用者新增至名為 **Admins** 的群組。

```
Add-IAMUserToGroup -UserName "Bob" -GroupName "Admins"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [AddUserToGroup](#)。

Disable-IAMMFADevice

以下程式碼範例顯示如何使用 Disable-IAMMFADevice。

PowerShell V4 的工具

範例 1：此命令會停用與序號為 **123456789012** 的使用者 **Bob** 關聯的硬體 MFA 裝置。

```
Disable-IAMMFADevice -UserName "Bob" -SerialNumber "123456789012"
```

範例 2：此命令會停用與 ARN 為 **arn:aws:iam::210987654321:mfa/David** 的使用者 **David** 關聯的虛擬 MFA 裝置。請注意，虛擬 MFA 裝置不會從帳戶中刪除。虛擬裝置仍然存在，並顯示在 **Get-IAMVirtualMFADevice** 命令的輸出中。必須先使用 **Remove-IAMVirtualMFADevice** 命令刪除舊的虛擬 MFA 裝置，才能為同一使用者建立新裝置。

```
Disable-IAMMFADevice -UserName "David" -SerialNumber "arn:aws:iam::210987654321:mfa/David"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeactivateMfaDevice](#)。

Edit-IAMPassword

以下程式碼範例顯示如何使用 Edit-IAMPassword。

PowerShell V4 的工具

範例 1：此命令會變更執行命令的使用者密碼。此命令只能由 IAM 使用者呼叫。如果您在使用 AWS 帳戶（根）登入資料登入時呼叫此命令，命令會傳回 **InvalidUserType** 錯誤。

```
Edit-IAMPASSWORD -OldPassword "MyOldP@ssw0rd" -NewPassword "MyNewP@ssw0rd"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ChangePassword](#)。

Enable-IAMMFADevice

以下程式碼範例顯示如何使用 Enable-IAMMFADevice。

PowerShell V4 的工具

範例 1：此命令會啟用序號為 **987654321098** 的硬體 MFA 裝置，並將裝置與使用者 **Bob** 關聯起來。它包含裝置中依序排列的前兩個代碼。

```
Enable-IAMMFADevice -UserName "Bob" -SerialNumber "987654321098" -  
AuthenticationCode1 "12345678" -AuthenticationCode2 "87654321"
```

範例 2：此範例會建立並啟用虛擬 MFA 裝置。第一個命令會建立虛擬裝置，並在變數 **\$MFADevice** 中傳回裝置的物件表示。可以使用 **.Base32StringSeed** 或 **QRCodePng** 屬性來設定使用者的軟體應用程式。最後一個命令會將裝置指派給使用者 **David**，依裝置序號標識裝置。命令也會依序包含來自虛擬 MFA 裝置的前兩個代碼，AWS 以將裝置與 同步。

```
$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  
-AuthenticationCode1 "24681357" -AuthenticationCode2  
"13572468"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [EnableMfaDevice](#)。

Get-IAMAccessKey

以下程式碼範例顯示如何使用 Get-IAMAccessKey。

PowerShell V4 的工具

範例 1：此命令會列出名為 **Bob** 之 IAM 使用者的存取金鑰。請注意，IAM 使用者的私密存取金鑰無法列出。如果私密存取金鑰遺失，必須使用 **New-IAMAccessKey** cmdlet 建立新的存取金鑰。

```
Get-IAMAccessKey -UserName "Bob"
```

輸出：

AccessKeyId	CreateDate	Status	UserName
-----	-----	-----	-----
AKIAIOSFODNN7EXAMPLE	12/3/2014 10:53:41 AM	Active	Bob
AKIAI44QH8DHBEXAMPLE	6/6/2013 8:42:26 PM	Inactive	Bob

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListAccessKeys](#)。

Get-IAMAccessKeyLastUsed

以下程式碼範例顯示如何使用 Get-IAMAccessKeyLastUsed。

PowerShell V4 的工具

範例 1：傳回所提供存取金鑰的擁有使用者名稱和上次使用資訊。

```
Get-IAMAccessKeyLastUsed -AccessKeyId ABCDEXAMPLE
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetAccessKeyLastUsed](#)。

Get-IAMAccountAlias

以下程式碼範例顯示如何使用 Get-IAMAccountAlias。

PowerShell V4 的工具

範例 1：此命令會傳回 AWS 帳戶的帳戶別名。

```
Get-IAMAccountAlias
```

輸出：

```
ExampleCo
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListAccountAliases](#)。

Get-IAMAccountAuthorizationDetail

以下程式碼範例顯示如何使用 Get-IAMAccountAuthorizationDetail。

PowerShell V4 的工具

範例 1：此範例會取得 AWS 帳戶中身分的授權詳細資訊，並顯示傳回物件的元素清單，包括使用者、群組和角色。例如，**UserDetailList** 屬性會顯示使用者的詳細資訊。**RoleDetailList** 和 **GroupDetailList** 屬性提供類似的資訊。

```
$Details=Get-IAMAccountAuthorizationDetail
$Details
```

輸出：

```
GroupDetailList : {Administrators, Developers, Testers, Backup}
IsTruncated     : False
Marker          :
RoleDetailList  : {TestRole1, AdminRole, TesterRole, clirole...}
UserDetailList  : {Administrator, Bob, BackupToS3, }
```

```
$Details.UserDetailList
```

輸出：

```
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}

```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [GetAccountAuthorizationDetails](#)。

Get-IAMAccountPasswordPolicy

以下程式碼範例顯示如何使用 Get-IAMAccountPasswordPolicy。

PowerShell V4 的工具

範例 1：此範例會傳回目前帳戶之密碼政策的詳細資訊。如果沒有為帳戶定義密碼政策，命令會傳回 **NoSuchEntity** 錯誤。

```
Get-IAMAccountPasswordPolicy
```

輸出：

```

AllowUsersToChangePassword : True
ExpirePasswords             : True
HardExpiry                  : False
MaxPasswordAge              : 90
MinimumPasswordLength       : 8
PasswordReusePrevention     : 20
RequireLowercaseCharacters  : True
RequireNumbers               : True
RequireSymbols               : False
RequireUppercaseCharacters  : True

```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [GetAccountPasswordPolicy](#)。

Get-IAMAccountSummary

以下程式碼範例顯示如何使用 Get-IAMAccountSummary。

PowerShell V4 的工具

範例 1：此範例會傳回 AWS 帳戶中目前 IAM 實體用量和目前 IAM 實體配額的相關資訊。

```
Get-IAMAccountSummary
```

輸出：

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
...	

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetAccountSummary](#)。

Get-IAMAttachedGroupPolicyList

以下程式碼範例顯示如何使用 Get-IAMAttachedGroupPolicyList。

PowerShell V4 的工具

範例 1：此命令會傳回連接到 **Admins** AWS 帳戶中名為 **Admins** 之 IAM 群組的受管政策的名稱和 ARNs。若要查看內嵌在群組中的內嵌政策的清單，請使用 **Get-IAMGroupPolicyList** 命令。

```
Get-IAMAttachedGroupPolicyList -GroupName "Admins"
```

輸出：

PolicyArn	PolicyName
-----	-----
arn:aws:iam::aws:policy/SecurityAudit	SecurityAudit
arn:aws:iam::aws:policy/AdministratorAccess	AdministratorAccess

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListAttachedGroupPolicies](#)。

Get-IAMAttachedRolePolicyList

以下程式碼範例顯示如何使用 Get-IAMAttachedRolePolicyList。

PowerShell V4 的工具

範例 1：此命令會傳回連接至 AWS 帳戶中名為 **SecurityAuditRole** 之 IAM 角色的受管政策的名稱和 ARN。若要查看內嵌在角色中的內嵌政策清單，請使用 **Get-IAMRolePolicyList** 命令。

```
Get-IAMAttachedRolePolicyList -RoleName "SecurityAuditRole"
```

輸出：

PolicyArn	PolicyName
-----	-----
arn:aws:iam::aws:policy/SecurityAudit	SecurityAudit

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListAttachedRolePolicies](#)。

Get-IAMAttachedUserPolicyList

以下程式碼範例顯示如何使用 Get-IAMAttachedUserPolicyList。

PowerShell V4 的工具

範例 1：此命令會傳回 **Bob** AWS 帳戶中名為 之 IAM 使用者之受管政策的名稱和 ARNs。若要查看內嵌在 IAM 使用者中的內嵌政策的清單，請使用 **Get-IAMUserPolicyList** 命令。

```
Get-IAMAttachedUserPolicyList -UserName "Bob"
```

輸出：

PolicyArn	PolicyName
-----	-----
arn:aws:iam::aws:policy/TesterPolicy	TesterPolicy

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListAttachedUserPolicies](#)。

Get-IAMContextKeysForCustomPolicy

以下程式碼範例顯示如何使用 Get-IAMContextKeysForCustomPolicy。

PowerShell V4 的工具

範例 1：此範例會擷取所提供政策 json 中的所有內容索引鍵。若要提供多個政策，可以採用以逗號分隔的值清單形式提供。

```
$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/"},"Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
Get-IAMContextKeysForCustomPolicy -PolicyInputList $policy1,$policy2
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GetContextKeysForCustomPolicy](#)。

Get-IAMContextKeysForPrincipalPolicy

以下程式碼範例顯示如何使用 Get-IAMContextKeysForPrincipalPolicy。

PowerShell V4 的工具

範例 1：此範例會擷取所提供政策 json 中的所有內容索引鍵，以及連接至 IAM 實體 (使用者/角色等) 的政策。對於 -PolicyInputList，您可以提供包含多個以逗號分隔的值的清單。

```
$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/"},"Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
Get-IAMContextKeysForPrincipalPolicy -PolicyInputList $policy1,$policy2 -
PolicySourceArn arn:aws:iam::852640994763:user/TestUser
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GetContextKeysForPrincipalPolicy](#)。

Get-IAMCredentialReport

以下程式碼範例顯示如何使用 Get-IAMCredentialReport。

PowerShell V4 的工具

範例 1：此範例會開啟傳回的報告，並以文字行陣列的形式將其輸出至管道。第一行是帶有以逗號分隔的資料欄名稱的標題。每個連續資料列是一個使用者的詳細資訊列，每個欄位以逗號分隔。必須先使用 **Request-IAMCredentialReport** cmdlet 產生報告，然後才能檢視報告。若要以單一字串擷取報告，請使用 **-Raw**，勿用 **-AsTextArray**。**-AsTextArray** 交換器也接受別名 **-SplitLines**。如需輸出中資料欄的完整清單，請參閱服務 API 參考。請注意，如果不使用 **-AsTextArray** 或 **-SplitLines**，必須使用 .NET **StreamReader** 類從 **.Content** 屬性擷取文字。

```
Request-IAMCredentialReport
```

輸出：

Description	State
-----	-----
No report exists. Starting a new report generation task	STARTED

```
Get-IAMCredentialReport -AsTextArray
```

輸出：

```
user,arn,user_creation_time,password_enabled,password_last_used,password_last_changed,password_last_changed,root_account,arn:aws:iam::123456789012:root,2014-10-15T16:31:25+00:00,not_supported,2015-04-20T16:06:00+00:00,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+00:00,A,false,true,2014-12-03T18:53:41+00:00,true,2015-03-25T20:38:14+00:00,false,N/A,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,A,false,false,N/A,false,N/A,false,2015-04-20T20:00:12+00:00,false,N/A
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetCredentialReport](#)。

Get-IAMEntitiesForPolicy

以下程式碼範例顯示如何使用 Get-IAMEntitiesForPolicy。

PowerShell V4 的工具

範例 1：此範例會傳回已連接政策 **arn:aws:iam::123456789012:policy/TestPolicy** 的 IAM 群組、角色和使用者的清單。

```
Get-IAMEntitiesForPolicy -PolicyArn "arn:aws:iam::123456789012:policy/TestPolicy"
```

輸出：

```
IsTruncated : False
Marker      :
PolicyGroups : {}
PolicyRoles : {testRole}
PolicyUsers  : {Bob, Theresa}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListEntitiesForPolicy](#)。

Get-IAMGroup

以下程式碼範例顯示如何使用 Get-IAMGroup。

PowerShell V4 的工具

範例 1：此範例會傳回 IAM 群組 **Testers** 的詳細資訊，包括屬於該群組的所有 IAM 使用者的集合。

```
$results = Get-IAMGroup -GroupName "Testers"
$results
```

輸出：

Group	IsTruncated	Marker
Users		
----	-----	-----

```
Amazon.IdentityManagement.Model.Group      False
    {Theresa, David}
```

```
$results.Group
```

輸出：

```
Arn          : arn:aws:iam::123456789012:group/Testers
CreateDate   : 12/10/2014 3:39:11 PM
GroupId      : 3RHNZZGQJ7QHMAEXAMPLE1
GroupName    : Testers
Path         : /
```

```
$results.Users
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetGroup](#)。

Get-IAMGroupForUser

以下程式碼範例顯示如何使用 Get-IAMGroupForUser。

PowerShell V4 的工具

範例 1：此範例會傳回 IAM 使用者 **David** 所屬之 IAM 群組的清單。

```
Get-IAMGroupForUser -UserName David
```

輸出：

```
Arn      : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId   : 6WCH4TRY3KIHIEEXAMPLE1
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      : /
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListGroupForUser](#)。

Get-IAMGroupList

以下程式碼範例顯示如何使用 Get-IAMGroupList。

PowerShell V4 的工具

範例 1：此範例會傳回目前 中定義之所有 IAM 群組的集合 AWS 帳戶。

```
Get-IAMGroupList
```

輸出：

```
Arn      : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId   : 6WCH4TRY3KIHIEEXAMPLE1
```

```

GroupName : Administrators
Path      : /

Arn       : arn:aws:iam::123456789012:group/Developers
CreateDate : 12/10/2014 3:38:55 PM
GroupId   : ZU2E0WMK6WBZ0EXAMPLE2
GroupName : Developers
Path      : /

Arn       : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId   : RHNZZGQJ7QHMAEXAMPLE3
GroupName : Testers
Path      : /

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListGroups](#)。

Get-IAMGroupPolicy

以下程式碼範例顯示如何使用 Get-IAMGroupPolicy。

PowerShell V4 的工具

範例 1：此範例會傳回群組 **PowerUserAccess-Testers** 名為 **Testers** 的嵌入式內嵌政策的詳細資訊。**PolicyDocument** 屬性經過 URL 編碼。在此範例中，它會使用 **UrlDecode** .NET 方法解碼。

```
$results = Get-IAMGroupPolicy -GroupName Testers -PolicyName PowerUserAccess-Testers
$results
```

輸出：

```

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": "*"   
  }  
]
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetGroupPolicy](#)。

Get-IAMGroupPolicyList

以下程式碼範例顯示如何使用 Get-IAMGroupPolicyList。

PowerShell V4 的工具

範例 1：此範例會傳回內嵌在群組 **Testers** 中的內嵌政策清單。若要取得連接到群組的受管政策，請使用 **Get-IAMAttachedGroupPolicyList** 命令。

```
Get-IAMGroupPolicyList -GroupName Testers
```

輸出：

```
Deny-Assume-S3-Role-In-Production  
PowerUserAccess-Testers
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListGroupPolicies](#)。

Get-IAMInstanceProfile

以下程式碼範例顯示如何使用 Get-IAMInstanceProfile。

PowerShell V4 的工具

範例 1：此範例會傳回目前 AWS 帳戶中定義的名為 **ec2instanceroles** 之執行個體描述檔的詳細資訊。

```
Get-IAMInstanceProfile -InstanceProfileName ec2instancerole
```

輸出：

```
Arn           : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate    : 2/17/2015 2:49:04 PM
InstanceId    : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path          : /
Roles         : {ec2instancerole}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [GetInstanceProfile](#)。

Get-IAMInstanceProfileForRole

以下程式碼範例顯示如何使用 Get-IAMInstanceProfileForRole。

PowerShell V4 的工具

範例 1：此範例會傳回與角色 **ec2instancerole** 關聯之執行個體設定檔的詳細資訊。

```
Get-IAMInstanceProfileForRole -RoleName ec2instancerole
```

輸出：

```
Arn           : arn:aws:iam::123456789012:instance-profile/
ec2instancerole
CreateDate    : 2/17/2015 2:49:04 PM
InstanceId    : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path          : /
Roles         : {ec2instancerole}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListInstanceProfilesForRole](#)。

Get-IAMInstanceProfileList

以下程式碼範例顯示如何使用 Get-IAMInstanceProfileList。

PowerShell V4 的工具

範例 1：此範例會傳回目前中定義的執行個體描述檔集合 AWS 帳戶。

```
Get-IAMInstanceProfileList
```

輸出：

```
Arn          : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate   : 2/17/2015 2:49:04 PM
InstanceId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path         : /
Roles        : {ec2instancerole}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListInstanceProfiles](#)。

Get-IAMLoginProfile

以下程式碼範例顯示如何使用 Get-IAMLoginProfile。

PowerShell V4 的工具

範例 1：此範例會傳回密碼建立日期，以及 IAM 使用者 **David** 是否需要重設密碼。

```
Get-IAMLoginProfile -UserName David
```

輸出：

CreateDate	PasswordResetRequired	UserName
-----	-----	-----
12/10/2014 3:39:44 PM	False	David

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [GetLoginProfile](#)。

Get-IAMMFADevice

以下程式碼範例顯示如何使用 Get-IAMMFADevice。

PowerShell V4 的工具

範例 1：此範例會傳回指派給 IAM 使用者 **David** 之 MFA 裝置的詳細資訊。在此範例中，您可以確定這是虛擬裝置，因為 **SerialNumber** 是 ARN，而不是實體裝置的實際序號。

```
Get-IAMMFADevice -UserName David
```

輸出：

EnableDate	SerialNumber	UserName
-----	-----	-----
4/8/2015 9:41:10 AM	arn:aws:iam::123456789012:mfa/David	David

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListMfaDevices](#)。

Get-IAMOpenIDConnectProvider

以下程式碼範例顯示如何使用 Get-IAMOpenIDConnectProvider。

PowerShell V4 的工具

範例 1：此範例會傳回 ARN 為 **arn:aws:iam::123456789012:oidc-provider/accounts.google.com** 的 OpenID Connect 提供者的詳細資訊。**ClientIDList** 屬性是一個集合，包含為此提供者定義的所有用戶端 ID。

```
Get-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
arn:aws:iam::123456789012:oidc-provider/oidc.example.com
```

輸出：

ClientIDList Url	CreateDate	ThumbprintList
-----	-----	-----

{MyOIDCApp}	2/3/2015 3:00:30 PM	
{12345abcdefghijk67890lmnopqrst98765uvwxy}		oidc.example.com

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetOpenIdConnectProvider](#)。

Get-IAMOpenIDConnectProviderList

以下程式碼範例顯示如何使用 Get-IAMOpenIDConnectProviderList。

PowerShell V4 的工具

範例 1：此範例會傳回目前的 AWS 帳戶中定義之所有 OpenID Connect 提供者的 ARNS 清單。

```
Get-IAMOpenIDConnectProviderList
```

輸出：

```
Arn
---
arn:aws:iam::123456789012:oidc-provider/server.example.com
arn:aws:iam::123456789012:oidc-provider/another.provider.com
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListOpenIdConnectProviders](#)。

Get-IAMPolicy

以下程式碼範例顯示如何使用 Get-IAMPolicy。

PowerShell V4 的工具

範例 1：此範例會傳回 ARN 為 **arn:aws:iam::123456789012:policy/MySamplePolicy** 之受管政策的詳細資訊。

```
Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GetPolicy](#)。

Get-IAMPolicyList

以下程式碼範例顯示如何使用 Get-IAMPolicyList。

PowerShell V4 的工具

範例 1：此範例會傳回目前 AWS 帳戶中可用的前三個受管政策的集合。由於 **-scope** 未指定，因此預設為 **all**，並同時包含 AWS 受管和客戶受管政策。

```
Get-IAMPolicyList -MaxItem 3
```

輸出：

```
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      : SULJS02FYVPYGEXAMPLE3
PolicyName    : AWSMarketplaceFullAccess
UpdateDate    : 2/11/2015 9:21:45 AM

```

範例 2：此範例會傳回目前 AWS 帳戶中可用的前兩個客戶受管政策的集合。它使用 **-Scope local** 將輸出限制為僅限客戶管理政策。

```
Get-IAMPolicyList -Scope local -MaxItem 2
```

輸出：

```

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/policyforec2instanceroles
AttachmentCount : 1
CreateDate    : 2/17/2015 2:51:38 PM
DefaultVersionId : v11
Description   :
IsAttachable  : True
Path          : /
PolicyId      : X5JPBLJH2Z2S0EXAMPLE5
PolicyName    : policyforec2instanceroles
UpdateDate    : 2/18/2015 8:52:31 AM

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListPolicies](#)。

Get-IAMPolicyVersion

以下程式碼範例顯示如何使用 Get-IAMPolicyVersion。

PowerShell V4 的工具

此範例會傳回 ARN 為 **arn:aws:iam::123456789012:policy/MyManagedPolicy** 之 v2 版本政策的政策文件。**Document** 屬性中的政策文件經過 URL 編碼，在此範例中使用 **UrlDecode** .NET 方法解碼。

```
$results = Get-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/
MyManagedPolicy -VersionId v2
$results
```

輸出：

```
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": "*"
  }
}
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetPolicyVersion](#)。

Get-IAMPolicyVersionList

以下程式碼範例顯示如何使用 Get-IAMPolicyVersionList。

PowerShell V4 的工具

範例 1：此範例會傳回 ARN 為 **arn:aws:iam::123456789012:policy/MyManagedPolicy** 之政策的可用版本清單。若要取得特定版本的策略文件，請使用 **Get-IAMPolicyVersion** 命令並指定想要文件的 **VersionId**。

```
Get-IAMPolicyVersionList -PolicyArn arn:aws:iam::123456789012:policy/MyManagedPolicy
```

輸出：

CreateDate VersionId	Document	IsDefaultVersion
----- -----	-----	-----
2/12/2015 9:39:53 AM v2		True
2/12/2015 9:39:09 AM v1		False

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListPolicyVersions](#)。

Get-IAMRole

以下程式碼範例顯示如何使用 **Get-IAMRole**。

PowerShell V4 的工具

範例 1：此範例會傳回 **lambda_exec_role** 的詳細資訊。其中包含指定誰可以擔任此角色的信任政策文件。政策文件經過 URL 編碼，可以使用 .NET **UrlDecode** 方法解碼。在此範例中，原始政策在上傳至政策之前已移除所有空格。若要查看決定擔任該角色之人員可以執行的動作的許可政策文件，請將 **Get-IAMRolePolicy** 用於內嵌政策，並將 **Get-IAMPolicyVersion** 用於連接的受管政策。

```
$results = Get-IamRole -RoleName lambda_exec_role
$results | Format-List
```

輸出：

```
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%7D
CreateDate                : 4/2/2015 9:16:11 AM
Path                      : /
RoleId                    : 2YBIKAIBHNKB4EXAMPLE1
RoleName                  : lambda_exec_role
```

```
$policy = [System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
$policy
```

輸出：

```
{"Version":"2012-10-17","Statement":[{"Sid":"","Effect":"Allow","Principal":{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [GetRole](#)。

Get-IAMRoleList

以下程式碼範例顯示如何使用 Get-IAMRoleList。

PowerShell V4 的工具

範例 1：此範例會擷取 AWS 帳戶中所有 IAM 角色的清單。

```
Get-IAMRoleList
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListRoles](#)。

Get-IAMRolePolicy

以下程式碼範例顯示如何使用 Get-IAMRolePolicy。

PowerShell V4 的工具

範例 1：此範例會傳回內嵌在 IAM 角色 **lambda_exec_role** 中的名為 **oneClick_lambda_exec_role_policy** 之政策的許可政策文件。產生的政策文件經過 URL 編碼。在此範例中，它會使用 **UrlDecode .NET** 方法解碼。

```
$results = Get-IAMRolePolicy -RoleName lambda_exec_role -PolicyName
oneClick_lambda_exec_role_policy
$results
```

輸出：

PolicyDocument	PolicyName
Username ----- ----- %7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%... oneClick_lambda_exec_role_policy lambda_exec_role	

```
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
```

輸出：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:*"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::*"
      ]
    }
  ]
}
```

```
]
}
]
}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GetRolePolicy](#)。

Get-IAMRolePolicyList

以下程式碼範例顯示如何使用 Get-IAMRolePolicyList。

PowerShell V4 的工具

範例 1：此範例會傳回內嵌在 IAM 角色 **lamda_exec_role** 中之內嵌政策的名稱清單。若要查看內嵌政策的詳細資訊，請使用 **Get-IAMRolePolicy** 命令。

```
Get-IAMRolePolicyList -RoleName lambda_exec_role
```

輸出：

```
oneClick_lambda_exec_role_policy
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListRolePolicies](#)。

Get-IAMRoleTagList

以下程式碼範例顯示如何使用 Get-IAMRoleTagList。

PowerShell V4 的工具

範例 1：此範例會擷取與角色關聯的標籤。

```
Get-IAMRoleTagList -RoleName MyRoleName
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListRoleTags](#)。

Get-IAMSAMLProvider

以下程式碼範例顯示如何使用 Get-IAMSAMLProvider。

PowerShell V4 的工具

範例 1：此範例會擷取 ARM 為 `arn:aws:iam::123456789012:saml-provider/SAMLADFS` 的 SAML 2.0 提供者的詳細資訊。回應包含您從身分提供者取得以建立 AWS SAML 提供者實體的中繼資料文件，以及建立和過期日期。

```
Get-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS
```

輸出：

CreateDate ValidUntil	SAMLMetadataDocument
-----	-----
12/23/2014 12:16:55 PM 12/23/2114 12:16:54 PM	<EntityDescriptor ID="_12345678-1234-5678-9012-example1...

- 如需 API 詳細資訊，請參閱 [AWS Tools for PowerShell Cmdlet 參考 \(V4\)](#) 中的 [GetSamlProvider](#)。

Get-IAMSAMLProviderList

以下程式碼範例顯示如何使用 `Get-IAMSAMLProviderList`。

PowerShell V4 的工具

範例 1：此範例會擷取在目前 AWS 帳戶中建立的 SAML 2.0 提供者的清單。它會傳回每個 SAML 提供者的 ARN、建立日期和過期日期。

```
Get-IAMSAMLProviderList
```

輸出：

Arn ValidUntil	CreateDate
---	-----
arn:aws:iam::123456789012:saml-provider/SAMLADFS 12/23/2114 12:16:54 PM	12/23/2014 12:16:55 PM

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListSAMLProviders](#)。

Get-IAMServerCertificate

以下程式碼範例顯示如何使用 Get-IAMServerCertificate。

PowerShell V4 的工具

範例 1：此範例會擷取名為 **MyServerCertificate** 之伺服器憑證的詳細資訊。可以在 **CertificateBody** 和 **ServerCertificateMetadata** 屬性中找到憑證詳細資訊。

```
$result = Get-IAMServerCertificate -ServerCertificateName MyServerCertificate
$result | format-list
```

輸出：

```
CertificateBody          : -----BEGIN CERTIFICATE-----

MIICiTCCAfICCQD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMaKGA1UEBhMC

VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6

b24xFDASBgNVBAstTC01BTSBDb25zb2x1MRIwEAYDVQQDEw1UZXRhbnQ2sYWMxHzAd

BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

MTIwNDI0MjA0NTIxWjCBiDELMaKGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYD

VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAstTC01BTSBDb25z

b2x1MRIwEAYDVQQDEw1UZXRhbnQ2sYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt

YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn

+a4GmWIWJ

21uUSfwfEvySwTC2XADZ4nB+BLygVIk60CpiwsZ3G93vUEI03IyNoH/

f0wYK8m9T

rDHudUZg3qX4waLG5M43q7Wgc/

MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE

Ibb30hjZnzcVQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4

nUhVVxYUntneD9+h8Mg9q6q

+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
```

```

FFBjvSfpJI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----
CertificateChain      :
ServerCertificateMetadata :
Amazon.IdentityManagement.Model.ServerCertificateMetadata

```

```
$result.ServerCertificateMetadata
```

輸出：

```

Arn                : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration          : 1/14/2018 9:52:36 AM
Path                : /Org1/Org2/
ServerCertificateId : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
UploadDate          : 4/21/2015 11:14:16 AM

```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetServerCertificate](#)。

Get-IAMServerCertificateList

以下程式碼範例顯示如何使用 Get-IAMServerCertificateList。

PowerShell V4 的工具

範例 1：此範例會擷取已上傳至目前 AWS 帳戶的伺服器憑證的清單。

```
Get-IAMServerCertificateList
```

輸出：

```

Arn                : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration          : 1/14/2018 9:52:36 AM
Path                : /Org1/Org2/
ServerCertificateId : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate

```

```
UploadDate           : 4/21/2015 11:14:16 AM
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListServerCertificates](#)。

Get-IAMServiceLastAccessedDetail

以下程式碼範例顯示如何使用 Get-IAMServiceLastAccessedDetail。

PowerShell V4 的工具

範例 1：此範例提供請求呼叫中關聯之 IAM 實體 (使用者、群組、角色或政策) 上次存取的服務詳細資訊。

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

輸出：

```
f0b7a819-eab0-929b-dc26-ca598911cb9f
```

```
Get-IAMServiceLastAccessedDetail -JobId f0b7a819-eab0-929b-dc26-ca598911cb9f
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetServiceLastAccessedDetails](#)。

Get-IAMServiceLastAccessedDetailWithEntity

以下程式碼範例顯示如何使用 Get-IAMServiceLastAccessedDetailWithEntity。

PowerShell V4 的工具

範例 1：此範例提供在請求中，該個別 IAM 實體上次存取服務的時間戳記。

```
$results = Get-IAMServiceLastAccessedDetailWithEntity -JobId f0b7a819-eab0-929b-dc26-ca598911cb9f -ServiceNamespace ec2
$results
```

輸出：

```
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
```

輸出：

```
EntityInfo                               LastAuthenticated
-----
Amazon.IdentityManagement.Model.EntityInfo 11/16/19 3:47:00 PM
```

```
$results.EntityInfo
```

輸出：

```
Arn  : arn:aws:iam::123456789012:user/TestUser
Id   : AIDA4NBK5CXF5TZHU1234
Name : TestUser
Path : /
Type : USER
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GetServiceLastAccessedDetailsWithEntities](#)。

Get-IAMSigningCertificate

以下程式碼範例顯示如何使用 Get-IAMSigningCertificate。

PowerShell V4 的工具

範例 1：此範例擷取與名為 **Bob** 之使用者關聯的簽署憑證的詳細資訊。

```
Get-IAMSigningCertificate -UserName Bob
```

輸出：

```

CertificateBody : -----BEGIN CERTIFICATE-----
                  MIICiTCCAfICCQD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
                  VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
                  b24xFDASBgNVBAstC01BTSBDb25zb2x1MRIwEAYDVQQDEw1UZXRhbnQ2LsYWMxHAd
                  BgkqhkiG9w0BCQEWEG5vb25lQGFTYXpvi5jb20wHhcNMTEwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYD
                  VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAstC01BTSBDb25z
                  b2x1MRIwEAYDVQQDEw1UZXRhbnQ2LsYWMxHAdBgkqhkiG9w0BCQEWEG5vb25lQGFT
                  YXpvi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMak0dn+a4GmWIWJ
                  21uUSfwfEvySwTc2XADZ4nB+BLygVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
                  rDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
                  Ibb30hjZnzcvcQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                  nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwax1Aoo7TJHidbtS4J5iNmZgXL0Fkb
                  FFBjvSfpJI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjSTb
                  NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                  -----END CERTIFICATE-----

CertificateId   : Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
Status          : Active
UploadDate     : 4/20/2015 1:26:01 PM
UserName       : Bob

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListSigningCertificates](#)。

Get-IAMUser

以下程式碼範例顯示如何使用 Get-IAMUser。

PowerShell V4 的工具

範例 1：此範例會擷取名為 **David** 之使用者的詳細資訊。

```
Get-IAMUser -UserName David
```

輸出：

```

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
```

範例 2：此範例會擷取目前登入之 IAM 使用者的詳細資訊。

```
Get-IAMUser
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GetUser](#)。

Get-IAMUserList

以下程式碼範例顯示如何使用 Get-IAMUserList。

PowerShell V4 的工具

範例 1：此範例會擷取目前 中的使用者集合 AWS 帳戶。

```
Get-IAMUserList
```

輸出：

```
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        : L3EWNNDOM3YUEXAMPLE2
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

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListUsers](#)。

Get-IAMUserPolicy

以下程式碼範例顯示如何使用 Get-IAMUserPolicy。

PowerShell V4 的工具

範例 1：此範例會擷取內嵌在名為 **David** 之 IAM 使用者中的內嵌政策 **Davids_IAM_Admin_Policy** 的詳細資訊。政策文件經過 URL 編碼。

```
$results = Get-IAMUserPolicy -PolicyName Davids_IAM_Admin_Policy -UserName David
$results
```

輸出：

```

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": [

```

```
        "*"
    ]
}
]
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GetUserPolicy](#)。

Get-IAMUserPolicyList

以下程式碼範例顯示如何使用 Get-IAMUserPolicyList。

PowerShell V4 的工具

範例 1：此範例會擷取內嵌在名為 **David** 之 IAM 使用者中的內嵌政策的名稱清單。

```
Get-IAMUserPolicyList -UserName David
```

輸出：

```
Dauids_IAM_Admin_Policy
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListUserPolicies](#)。

Get-IAMUserTagList

以下程式碼範例顯示如何使用 Get-IAMUserTagList。

PowerShell V4 的工具

範例 1：此範例會擷取與使用者關聯的標籤。

```
Get-IAMUserTagList -UserName joe
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListUserTags](#)。

Get-IAMVirtualMFADevice

以下程式碼範例顯示如何使用 Get-IAMVirtualMFADevice。

PowerShell V4 的工具

範例 1：此範例會擷取指派給 AWS 帳戶中使用者的虛擬 MFA 裝置集合。每個裝置的 **User** 屬性是一個物件，包含裝置被分派給的 IAM 使用者的詳細資訊。

```
Get-IAMVirtualMFADevice -AssignmentStatus Assigned
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListVirtualMfaDevices](#)。

New-IAMAccessKey

以下程式碼範例顯示如何使用 New-IAMAccessKey。

PowerShell V4 的工具

範例 1：此範例會建立新的存取金鑰和私密存取金鑰對，並將其指派給使用者 **David**。請確定將 **AccessKeyId** 和 **SecretAccessKey** 值儲存到檔案中，因為這是唯一可以取得 **SecretAccessKey** 的時機。您稍後便無法擷取它。若您遺失了密碼金鑰，必須建立新的存取金鑰對。

```
New-IAMAccessKey -UserName David
```

輸出：

```
AccessKeyId      : AKIAIOSFODNN7EXAMPLE
CreateDate       : 4/13/2015 1:00:42 PM
```

```
SecretAccessKey : wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Status          : Active
UserName        : David
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateAccessKey](#)。

New-IAMAccountAlias

以下程式碼範例顯示如何使用 New-IAMAccountAlias。

PowerShell V4 的工具

範例 1：此範例會將您的帳戶 AWS 的帳戶別名變更為 **mycompanyaws**。使用者登入頁面的網址會變更為 <https://mycompanyaws.signin.aws.amazon.com/console>。使用您的帳戶 ID 號碼而非別名的原始 URL (<https://<accountidnumber>.signin.aws.amazon.com/console>) 繼續有效。但是，任何先前定義的別名型 URL 已失效。

```
New-IAMAccountAlias -AccountAlias mycompanyaws
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateAccountAlias](#)。

New-IAMGroup

以下程式碼範例顯示如何使用 New-IAMGroup。

PowerShell V4 的工具

範例 1：此範例會建立名為 **Developers** 的新 IAM 群組。

```
New-IAMGroup -GroupName Developers
```

輸出：

```
Arn          : arn:aws:iam::123456789012:group/Developers
CreateDate   : 4/14/2015 11:21:31 AM
GroupId      : QNEJ5PM4NFSQCEXAMPLE1
GroupName    : Developers
Path         : /
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateGroup](#)。

New-IAMInstanceProfile

以下程式碼範例顯示如何使用 New-IAMInstanceProfile。

PowerShell V4 的工具

範例 1：此範例會建立名為 **ProfileForDevEC2Instance** 的新 IAM 執行個體設定檔。必須單獨執行 **Add-IAMRoleToInstanceProfile** 命令，將執行個體設定檔與為執行個體提供許可的現有 IAM 角色關聯起來。最後，在啟動執行個體時，將執行個體設定檔連接到 EC2 執行個體。為此，請使用含 **InstanceProfile_Arn** 或 **InstanceProfile_Name** 參數的 **New-EC2Instance** cmdlet。

```
New-IAMInstanceProfile -InstanceProfileName ProfileForDevEC2Instance
```

輸出：

```
Arn                : arn:aws:iam::123456789012:instance-profile/
ProfileForDevEC2Instance
CreateDate         : 4/14/2015 11:31:39 AM
InstanceProfileId  : DYMFXL556EY46EXAMPLE1
InstanceProfileName : ProfileForDevEC2Instance
Path              : /
Roles              : {}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateInstanceProfile](#)。

New-IAMLoginProfile

以下程式碼範例顯示如何使用 New-IAMLoginProfile。

PowerShell V4 的工具

範例 1：此範例會為名為 Bob 的 IAM 使用者建立 (臨時) 密碼並設定在 **Bob** 下次登入時要求使用者變更密碼的旗標。

```
New-IAMLoginProfile -UserName Bob -Password P@ssw0rd -PasswordResetRequired $true
```


輸出：

CreateDate	PasswordResetRequired	UserName
-----	-----	-----
4/14/2015 12:26:30 PM	True	Bob

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateLoginProfile](#)。

New-IAMOpenIDConnectProvider

以下程式碼範例顯示如何使用 New-IAMOpenIDConnectProvider。

PowerShell V4 的工具

範例 1：此範例會建立與 URL **https://example.oidcprovider.com** 和用戶端 ID **my-testapp-1** 中的 OIDC 相容提供者服務關聯的 IAM OIDC 提供者。OIDC 提供者會提供指紋。若要驗證指紋，請依照 <http://docs.aws.amazon.com/IAM/latest/UserGuide/identity-providers-oidc-obtain-thumbprint.html> 中的步驟進行。

```
New-IAMOpenIDConnectProvider -Url https://example.oidcprovider.com -ClientIDList my-testapp-1 -ThumbprintList 990F419EXAMPLEECF12DDEDA5EXAMPLE52F20D9E
```

輸出：

```
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateOpenIdConnectProvider](#)。

New-IAMPolicy

以下程式碼範例顯示如何使用 New-IAMPolicy。

PowerShell V4 的工具

範例 1：此範例會在名為 **MySamplePolicy** 的目前 AWS 帳戶中建立新的 IAM **MySamplePolicy** 政策。檔案 **MySamplePolicy.json** 會提供政策內容。請注意，必須使用 **-Raw** 切換參數，才能成功處理 JSON 政策檔案。

```
New-IAMPolicy -PolicyName MySamplePolicy -PolicyDocument (Get-Content -Raw
MySamplePolicy.json)
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreatePolicy](#)。

New-IAMPolicyVersion

以下程式碼範例顯示如何使用 New-IAMPolicyVersion。

PowerShell V4 的工具

範例 1：此範例會建立新 v2 版的 IAM 政策 (其 ARN 為 **arn:aws:iam::123456789012:policy/MyPolicy**)，並將該版本設為預設版本。**NewPolicyVersion.json** 檔案會提供政策內容。請注意，必須使用 **-Raw** 切換參數，才能成功處理 JSON 政策檔案。

```
New-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -
PolicyDocument (Get-content -Raw NewPolicyVersion.json) -SetAsDefault $true
```

輸出：

CreateDate VersionId	Document	IsDefaultVersion
-----	-----	-----

4/15/2015 10:54:54 AM
v2

True

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreatePolicyVersion](#)。

New-IAMRole

以下程式碼範例顯示如何使用 New-IAMRole。

PowerShell V4 的工具

範例 1：此範例會建立名為 **MyNewRole** 的新角色，並將其連接至在 **NewRoleTrustPolicy.json** 檔案中找到的政策。請注意，必須使用 **-Raw** 切換參數，才能成功處理 JSON 政策檔案。輸出中顯示的政策文件是以 URL 編碼。在此範例中，它會使用 **UrlDecode** .NET 方法解碼。

```
$results = New-IAMRole -AssumeRolePolicyDocument (Get-Content -raw
NewRoleTrustPolicy.json) -RoleName MyNewRole
$results
```

輸出：

```
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%22Sid%22%3A%20%22%22%2C%0D%0A%20%20%20%20%20%20%22Effect%22%3A%20%22Allow%22%2C%0D%0A%20%20%20%20%20%20%22Principal%22%3A%20%7B%0D%0A%20%20%20%20%20%20%22AWS%22%3A%20%22arn%3Aaws%3Aiam%3A%3A123456789012%3ADavid%22%0D%0A%20%20%20%20%20%20%7D%2C%0D%0A%20%20%20%20%20%20%22Action%22%3A%20%22sts%3AAssumeRole%22%0D%0A%20%20%20%20%7D%0D%0A%20%20%5D%0D%0A%7D
CreateDate         : 4/15/2015 11:04:23 AM
Path               : /
RoleId            : V5PAJI2KPN4EAEXAMPLE1
RoleName          : MyNewRole
```

```
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:David"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateRole](#)。

New-IAMSAMLProvider

以下程式碼範例顯示如何使用 New-IAMSAMLProvider。

PowerShell V4 的工具

範例 1：此範例會在 IAM 中建立新的 SAML 提供者實體。它被命名為 **MySAMLProvider**，由在檔案 **SAMLMetaData.xml** 中找到的 SAML 中繼資料文件描述，該檔案需從 SAML 服務提供者的網站單獨下載。

```
New-IAMSAMLProvider -Name MySAMLProvider -SAMLMetadataDocument (Get-Content -Raw
SAMLMetaData.xml)
```

輸出：

```
arn:aws:iam::123456789012:saml-provider/MySAMLProvider
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateSAMLProvider](#)。

New-IAMServiceLinkedRole

以下程式碼範例顯示如何使用 New-IAMServiceLinkedRole。

PowerShell V4 的工具

範例 1：此範例會建立用於自動擴展服務的服務連結角色。

```
New-IAMServiceLinkedRole -AWSServiceName autoscaling.amazonaws.com -CustomSuffix
RoleNameEndsWithThis -Description "My service-linked role to support autoscaling"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateServiceLinkedRole](#)。

New-IAMUser

以下程式碼範例顯示如何使用 New-IAMUser。

PowerShell V4 的工具

範例 1：此範例會建立名為 **Bob** 的 IAM 使用者。如果 Bob 需要登入 AWS 主控台，則您必須分別執行命令 **New-IAMLoginProfile**，以使用密碼建立登入設定檔。如果 Bob 需要執行 AWS PowerShell 或跨平台 CLI 命令或進行 AWS API 呼叫，則必須單獨執行 **New-IAMAccessKey** 命令來建立存取金鑰。

```
New-IAMUser -UserName Bob
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [CreateUser](#)。

New-IAMVirtualMFADevice

以下程式碼範例顯示如何使用 New-IAMVirtualMFADevice。

PowerShell V4 的工具

範例 1：此範例會建立新的虛擬 MFA 裝置。第 2 行和第 3 行擷取虛擬 MFA 軟體程式建立帳戶所需的 **Base32StringSeed** 值 (做為 QR 程式碼的替代方案)。使用值設定程式後，依序從程式取得兩個驗證碼。最後，使用最後一個命令將虛擬 MFA 裝置連結至 IAM 使用者 **Bob**，並使用兩個驗證碼同步帳戶。

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$SR = New-Object System.IO.StreamReader($Device.Base32StringSeed)
$base32stringseed = $SR.ReadToEnd()
$base32stringseed
CZWZMCQNW4DEXAMPLE3V0UGXJFZYSUW7EXAMPLECR4NJFD65GX2SLUDW2EXAMPLE
```

輸出：

```
-- 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
```

範例 2：此範例會建立新的虛擬 MFA 裝置。第 2 行和第 3 行會擷取 **QRCodePNG** 值，並將其寫入檔案。此映像可由虛擬 MFA 軟體程式掃描，以建立帳戶 (做為手動輸入 Base32StringSeed 值的替代方案)。在虛擬 MFA 程式中建立帳戶後，請依序取得兩個驗證碼，並在最後一個命令中輸入它們，將虛擬 MFA 裝置連結至 IAM 使用者 **Bob** 並同步帳戶。

```
$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
```

輸出：

```
-- Pause here to scan PNG with virtual MFA program to register account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateVirtualMfaDevice](#)。

Publish-IAMServerCertificate

以下程式碼範例顯示如何使用 Publish-IAMServerCertificate。

PowerShell V4 的工具

範例 1：此範例會將新伺服器憑證上傳至 IAM 帳戶。包含憑證內文、私有金鑰和 (可選) 憑證鏈的檔案全都必須採用 PEM 編碼。請注意，參數需要檔案的實際內容，而不是檔案名稱。必須使用 **-Raw** 切換參數，才能成功處理檔案內容。

```
Publish-IAMServerCertificate -ServerCertificateName MyTestCert -CertificateBody
(Get-Content -Raw server.crt) -PrivateKey (Get-Content -Raw server.key)
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UploadServerCertificate](#)。

Publish-IAMSigningCertificate

以下程式碼範例顯示如何使用 Publish-IAMSigningCertificate。

PowerShell V4 的工具

範例 1：此範例會上傳新的 X.509 簽署憑證，並將其與名為 **Bob** 的 IAM 使用者關聯起來。包含憑證內文的檔案經過 PEM 編碼。**CertificateBody** 參數需要憑證檔案的實際內容，而不是檔案名稱。必須使用 **-Raw** 切換參數才能成功處理檔案。

```
Publish-IAMSigningCertificate -UserName Bob -CertificateBody (Get-Content -Raw
SampleSigningCert.pem)
```

輸出：

```
CertificateBody : -----BEGIN CERTIFICATE-----
```

```

MIICiTCCAFICCD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBA5TC0lBTSBDb25zb2x1MRIwEAYDVQQDEw1UZXR0Q21sYWxhZAd
BgkqhkiG9w0BCQEWEG5vb25lQGfYXpvi5jb20wHhCNMTEwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYD
VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBA5TC0lBTSBDb25z
b2x1MRIwEAYDVQQDEw1UZXR0Q21sYWxhZAdBgkqhkiG9w0BCQEWEG5vb25lQGfY
YXpvi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySWtC2XADZ4nB+BLyGVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb30hjZnzcVQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUHVvYUntneD9+h8Mg9q6q+auNKyExzyLwax1Aoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----

```

```

CertificateId      : Y3EK7RMEXAMPLESV33FCEXAMPLEHMLU
Status             : Active
UploadDate        : 4/20/2015 1:26:01 PM
UserName          : Bob

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [UploadSigningCertificate](#)。

Register-IAMGroupPolicy

以下程式碼範例顯示如何使用 Register-IAMGroupPolicy。

PowerShell V4 的工具

範例 1：此範例會將名為 **TesterPolicy** 的客戶管理政策連接到 IAM 群組 **Testers**。該群組中的使用者會立即受到該政策預設版本中定義之許可影響。

```

Register-IAMGroupPolicy -GroupName Testers -PolicyArn
arn:aws:iam::123456789012:policy/TesterPolicy

```

範例 2：此範例會將名為 **AdministratorAccess** 的 AWS 受管政策連接至 **AdministratorAccess** IAM 群組 **Admins**。該群組中的使用者會立即受到該政策最新版本中定義之許可影響。

```

Register-IAMGroupPolicy -GroupName Admins -PolicyArn arn:aws:iam::aws:policy/
AdministratorAccess

```


- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [AttachGroupPolicy](#)。

Register-IAMRolePolicy

以下程式碼範例顯示如何使用 Register-IAMRolePolicy。

PowerShell V4 的工具

範例 1：此範例會將名為 **SecurityAudit** 的 AWS 受管政策連接至 **CoSecurityAuditors** IAM 角色。該最新版本政策中所定義的許可會立即影響擔任該角色的使用者。

```
Register-IAMRolePolicy -RoleName CoSecurityAuditors -PolicyArn  
arn:aws:iam::aws:policy/SecurityAudit
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [AttachRolePolicy](#)。

Register-IAMUserPolicy

以下程式碼範例顯示如何使用 Register-IAMUserPolicy。

PowerShell V4 的工具

範例 1：此範例會將名為 **AmazonCognitoPowerUser** 的 AWS 受管政策連接至 IAM 使用者 **Bob**。該最新版本政策中所定義的許可會立即影響使用者。

```
Register-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::aws:policy/  
AmazonCognitoPowerUser
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [AttachUserPolicy](#)。

Remove-IAMAccessKey

以下程式碼範例顯示如何使用 Remove-IAMAccessKey。

PowerShell V4 的工具

範例 1：此範例 **AKIAIOSFODNN7EXAMPLE** 會從名為 的使用者中刪除具有金鑰 ID 的 AWS 存取金鑰對 **Bob**。

```
Remove-IAMAccessKey -AccessKeyId AKIAIOSFODNN7EXAMPLE -UserName Bob -Force
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteAccessKey](#)。

Remove-IAMAccountAlias

以下程式碼範例顯示如何使用 Remove-IAMAccountAlias。

PowerShell V4 的工具

範例 1：此範例會從 中移除帳戶別名 AWS 帳戶。在 https://mycompanyaws.signin.aws.amazon.com/console，使用者無法再使用別名登入頁面。您必須改為在 https://https://<accountidnumber>.signin.aws.amazon.com/console 使用原始 URL 搭配您的 AWS 帳戶 ID 號碼。

```
Remove-IAMAccountAlias -AccountAlias mycompanyaws
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteAccountAlias](#)。

Remove-IAMAccountPasswordPolicy

以下程式碼範例顯示如何使用 Remove-IAMAccountPasswordPolicy。

PowerShell V4 的工具

範例 1：此範例會刪除 的密碼政策，AWS 帳戶 並將所有值重設為其原始預設值。如果密碼政策目前不存在，則會出現下列錯誤訊息：找不到名為 PasswordPolicy 的帳戶政策。

```
Remove-IAMAccountPasswordPolicy
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteAccountPasswordPolicy](#)。

Remove-IAMClientIDFromOpenIDConnectProvider

以下程式碼範例顯示如何使用 Remove-IAMClientIDFromOpenIDConnectProvider。

PowerShell V4 的工具

範例 1：此範例會從與 ARN 為 **arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com** 的 IAM OIDC 提供者關聯的用戶端 ID 清單中移除用戶端 ID **My-TestApp-3**。

```
Remove-IAMClientIDFromOpenIDConnectProvider -ClientID My-TestApp-3  
-OpenIDConnectProviderArn arn:aws:iam::123456789012:oidc-provider/  
example.oidcprovider.com
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [RemoveClientIDFromOpenIDConnectProvider](#)。

Remove-IAMGroup

以下程式碼範例顯示如何使用 Remove-IAMGroup。

PowerShell V4 的工具

範例 1：此範例會刪除名為 **MyTestGroup** 的 IAM 群組。第一個命令會移除所有屬於群組的 IAM 使用者，第二個命令則會刪除 IAM 群組。這兩個命令皆可在沒有任何確認提示的情況下運作。

```
(Get-IAMGroup -GroupName MyTestGroup).Users | Remove-IAMUserFromGroup -GroupName  
MyTestGroup -Force  
Remove-IAMGroup -GroupName MyTestGroup -Force
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteGroup](#)。

Remove-IAMGroupPolicy

以下程式碼範例顯示如何使用 Remove-IAMGroupPolicy。

PowerShell V4 的工具

範例 1：此範例會從 IAM 群組 **Testers** 中移除名為 **TesterPolicy** 的內嵌政策。該群組中的使用者會立即失去此政策中定義的許可。

```
Remove-IAMGroupPolicy -GroupName Testers -PolicyName TestPolicy
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteGroupPolicy](#)。

Remove-IAMInstanceProfile

以下程式碼範例顯示如何使用 Remove-IAMInstanceProfile。

PowerShell V4 的工具

範例 1：此範例會刪除名為 **MyAppInstanceProfile** 的 EC2 執行個體設定檔。第一個命令將從執行個體設定檔分離任何角色，然後第二個命令將刪除執行個體設定檔。

```
(Get-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile).Roles | Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyAppInstanceProfile  
Remove-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteInstanceProfile](#)。

Remove-IAMLoginProfile

以下程式碼範例顯示如何使用 Remove-IAMLoginProfile。

PowerShell V4 的工具

範例 1：此範例會從名為 **Bob** 的 IAM 使用者中刪除登入設定檔。這可防止使用者登入 AWS 主控台。它不會阻止使用者使用可能仍連接至使用者帳戶的 AWS 存取金鑰來執行任何 AWS CLI、PowerShell 或 API 呼叫。

```
Remove-IAMLoginProfile -UserName Bob
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteLoginProfile](#)。

Remove-IAMOpenIDConnectProvider

以下程式碼範例顯示如何使用 Remove-IAMOpenIDConnectProvider。

PowerShell V4 的工具

範例 1：此範例會刪除連線至提供者 **example.oidcprovider.com** 的 IAM OIDC 提供者。請確定更新或刪除在角色信任政策的 **Principal** 元素中引用此提供者的任何角色。

```
Remove-IAMOpenIDConnectProvider -OpenIDConnectProviderArn  
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteOpenIdConnectProvider](#)。

Remove-IAMPolicy

以下程式碼範例顯示如何使用 Remove-IAMPolicy。

PowerShell V4 的工具

範例 1：此範例會刪除 ARN 為 **arn:aws:iam::123456789012:policy/MySamplePolicy** 的政策。在刪除政策之前，必須先執行 **Remove-IAMPolicyVersion** 刪除除預設版本之外的所有版本。還必須將政策與任何 IAM 使用者、群組或角色分離。

```
Remove-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

範例 2：此範例會刪除政策，方法是先刪除所有非預設政策版本，將其與所有連接的 IAM 實體分離，最後刪除政策本身。第一行擷取政策物件。第二行擷取未標記為集合預設版本的所有政策版本，然後刪除集合中的每個政策。第三行擷取連接政策的所有 IAM 使用者、群組和角色。第四行到第六行將政策與每個連接的實體分離。最後一行使用此命令移除受管政策以及剩餘的預設版本。此範例會在任何需要 **-Force** 切換參數的行中加入該參數，以抑制確認提示。

```
$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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeletePolicy](#)。

Remove-IAMPolicyVersion

以下程式碼範例顯示如何使用 Remove-IAMPolicyVersion。

PowerShell V4 的工具

範例 1：此範例會從 ARN 為 **arn:aws:iam::123456789012:policy/MySamplePolicy** 的政策中刪除標識為 **v2** 的版本。

```
Remove-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy -
VersionID v2
```

範例 2：此範例會先刪除所有非預設政策版本，然後刪除政策本身，從而刪除政策。第一行擷取政策物件。第二行擷取未標記為集合預設值的所有政策版本，然後使用此命令刪除集合中的每個政策。最後一行移除政策本身以及剩餘的預設版本。請注意，若要成功刪除受管政策，還必須使用 **Unregister-IAMUserPolicy**、**Unregister-IAMGroupPolicy** 和 **Unregister-IAMRolePolicy** 命令，將政策與任何使用者、群組或角色分離。請參閱 **Remove-IAMPolicy cmdlet** 的範例。

```
$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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeletePolicyVersion](#)。

Remove-IAMRole

以下程式碼範例顯示如何使用 Remove-IAMRole。

PowerShell V4 的工具

範例 1：此範例會從目前的 IAM 帳戶刪除名為 **MyNewRole** 的角色。在刪除角色之前，必須先使用 **Unregister-IAMRolePolicy** 命令來分離任何受管政策。內嵌政策會與角色一起刪除。

```
Remove-IAMRole -RoleName MyNewRole
```

範例 2：此範例會從名為 **MyNewRole** 的角色分離任何受管政策，然後刪除該角色。第一行將連接至角色的任何受管政策擷取為集合，然後讓集合中的每個政策與角色分離。第二行刪除角色本身。內嵌政策會與角色一起刪除。

```
Get-IAMAttachedRolePolicyList -RoleName MyNewRole | Unregister-IAMRolePolicy -  
RoleName MyNewRole  
Remove-IAMRole -RoleName MyNewRole
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteRole](#)。

Remove-IAMRoleFromInstanceProfile

以下程式碼範例顯示如何使用 Remove-IAMRoleFromInstanceProfile。

PowerShell V4 的工具

範例 1：此範例會將名為 **MyNewRole** 的角色從名為 **MyNewRole** 的 EC2 執行個體設定檔中刪除。在 IAM 主控台中建立的執行個體設定檔一律與角色同名，如本範例中所示。如果您在 API 或 CLI 中建立它們，則它們可以有不同的名稱。

```
Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyNewRole -RoleName MyNewRole  
-Force
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [RemoveRoleFromInstanceProfile](#)。

Remove-IAMRolePermissionsBoundary

以下程式碼範例顯示如何使用 Remove-IAMRolePermissionsBoundary。

PowerShell V4 的工具

範例 1：此範例示範如何移除連接至 IAM 角色的許可界限。

```
Remove-IAMRolePermissionsBoundary -RoleName MyRoleName
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteRolePermissionsBoundary](#)。

Remove-IAMRolePolicy

以下程式碼範例顯示如何使用 Remove-IAMRolePolicy。

PowerShell V4 的工具

範例 1：此範例會刪除內嵌在 IAM 角色 **S3BackupRole** 中的內嵌政策 **S3AccessPolicy**。

```
Remove-IAMRolePolicy -PolicyName S3AccessPolicy -RoleName S3BackupRole
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteRolePolicy](#)。

Remove-IAMRoleTag

以下程式碼範例顯示如何使用 Remove-IAMRoleTag。

PowerShell V4 的工具

範例 1：此範例會從名為「MyRoleName」且標籤索引鍵為「abac」的角色中移除標籤。若要移除多個標籤，請提供以逗號分隔的標籤索引鍵清單。

```
Remove-IAMRoleTag -RoleName MyRoleName -TagKey "abac","xyzw"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [UntagRole](#)。

Remove-IAMSAMLProvider

以下程式碼範例顯示如何使用 Remove-IAMSAMLProvider。

PowerShell V4 的工具

範例 1：此範例會刪除 ARN 為 **arn:aws:iam::123456789012:saml-provider/SAMLADFSPProvider** 的 IAM SAML 2.0 提供者。

```
Remove-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/SAMLADFSPProvider
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteSAMLProvider](#)。

Remove-IAMServerCertificate

以下程式碼範例顯示如何使用 Remove-IAMServerCertificate。

PowerShell V4 的工具

範例 1：此範例會刪除名為 **MyServerCert** 的伺服器憑證。

```
Remove-IAMServerCertificate -ServerCertificateName MyServerCert
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteServerCertificate](#)。

Remove-IAMServiceLinkedRole

以下程式碼範例顯示如何使用 Remove-IAMServiceLinkedRole。

PowerShell V4 的工具

範例 1：此範例已經刪除服務連結的角色。請注意，如果服務仍在使用該角色，則此命令會導致失敗。

```
Remove-IAMServiceLinkedRole -RoleName  
AWSServiceRoleForAutoScaling_RoleNameEndsWithThis
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteServiceLinkedRole](#)。

Remove-IAMSigningCertificate

以下程式碼範例顯示如何使用 Remove-IAMSigningCertificate。

PowerShell V4 的工具

範例 1：此範例會從名為 **Bob** 的 IAM 使用者中刪除 ID 為 **Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU** 的簽署憑證。

```
Remove-IAMSigningCertificate -UserName Bob -CertificateId  
Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteSigningCertificate](#)。

Remove-IAMUser

以下程式碼範例顯示如何使用 Remove-IAMUser。

PowerShell V4 的工具

範例 1：此範例會刪除名為 **Bob** 的 IAM 使用者。

```
Remove-IAMUser -UserName Bob
```

範例 2：此範例會刪除名為 **Theresa** 的 IAM 使用者以及任何必須先刪除的元素。

```
$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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteUser](#)。

Remove-IAMUserFromGroup

以下程式碼範例顯示如何使用 Remove-IAMUserFromGroup。

PowerShell V4 的工具

範例 1：此範例會從 **Testers** 群組中移除 IAM 使用者 **Bob**。

```
Remove-IAMUserFromGroup -GroupName Testers -UserName Bob
```

範例 2：此範例會尋找 IAM 使用者 **Theresa** 所屬的任意群組，然後從這些群組中移除 **Theresa**。

```
$groups = Get-IAMGroupForUser -UserName Theresa
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName -
  UserName Theresa -Force }
```

範例 3：此範例顯示從 **Testers** 群組中移除 IAM 使用者 **Bob** 的替代方式。

```
Get-IAMGroupForUser -UserName Bob | Remove-IAMUserFromGroup -UserName Bob -GroupName
  Testers -Force
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [RemoveUserFromGroup](#)。

Remove-IAMUserPermissionsBoundary

以下程式碼範例顯示如何使用 Remove-IAMUserPermissionsBoundary。

PowerShell V4 的工具

範例 1：此範例示範如何移除連接到 IAM 使用者的許可界限。

```
Remove-IAMUserPermissionsBoundary -UserName joe
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteUserPermissionsBoundary](#)。

Remove-IAMUserPolicy

以下程式碼範例顯示如何使用 Remove-IAMUserPolicy。

PowerShell V4 的工具

範例 1：此範例會刪除內嵌在名為 **Bob** 之 IAM 使用者中的、名為 **AccessToEC2Policy** 的內嵌政策。

```
Remove-IAMUserPolicy -PolicyName AccessToEC2Policy -UserName Bob
```

範例 2：此範例會尋找內嵌在名為 **Theresa** 的 IAM 使用者中的所有內嵌政策，然後將其刪除。

```
$inlinepols = Get-IAMUserPolicies -UserName Theresa  
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName  
    Theresa -Force }
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteUserPolicy](#)。

Remove-IAMUserTag

以下程式碼範例顯示如何使用 Remove-IAMUserTag。

PowerShell V4 的工具

範例 1：此範例會從名為「joe」且標籤索引鍵為「abac」和「xyzw」的使用者中移除標籤。若要移除多個標籤，請提供以逗號分隔的標籤索引鍵清單。

```
Remove-IAMUserTag -UserName joe -TagKey "abac","xyzw"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [UntagUser](#)。

Remove-IAMVirtualMFADevice

以下程式碼範例顯示如何使用 Remove-IAMVirtualMFADevice。

PowerShell V4 的工具

範例 1：此範例會刪除 ARN 為 **arn:aws:iam::123456789012:mfa/bob** 的 IAM 虛擬 MFA 裝置。

```
Remove-IAMVirtualMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/bob
```

範例 2：此範例會檢查 IAM 使用者 Theresa 是否已獲分派 MFA 裝置。如果找到一個，系統會為 IAM 使用者停用裝置。如果裝置是虛擬裝置，亦將刪除。

```
$mfa = Get-IAMMFADevice -UserName Theresa
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -SerialNumber
        $mfa.SerialNumber }
}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteVirtualMfaDevice](#)。

Request-IAMCredentialReport

以下程式碼範例顯示如何使用 Request-IAMCredentialReport。

PowerShell V4 的工具

範例 1：此範例會請求產生新的報告，這可以每四小時執行一次。如果最後一個報告仍然是最新的，「狀態」欄位會顯示：**COMPLETE**。**Get-IAMCredentialReport** 可用於檢視已完成的報告。

```
Request-IAMCredentialReport
```

輸出：

Description	State
-----	-----
No report exists. Starting a new report generation task	STARTED

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GenerateCredentialReport](#)。

Request-IAServiceLastAccessedDetail

以下程式碼範例顯示如何使用 Request-IAServiceLastAccessedDetail。

PowerShell V4 的工具

範例 1：此範例等同於 GenerateServiceLastAccessedDetails API 的 cmdlet。這提供了任務 ID，可用於 Get-IAServiceLastAccessedDetail 和 Get-IAServiceLastAccessedDetailWithEntity

```
Request-IAServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GenerateServiceLastAccessedDetails](#)。

Set-IAMDefaultPolicyVersion

以下程式碼範例顯示如何使用 Set-IAMDefaultPolicyVersion。

PowerShell V4 的工具

範例 1：此範例會將 ARN 為 **arn:aws:iam::123456789012:policy/MyPolicy** 的政策的 v2 版本設定為預設的作用中版本。

```
Set-IAMDefaultPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -VersionId v2
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [SetDefaultPolicyVersion](#)。

Set-IAMRolePermissionsBoundary

以下程式碼範例顯示如何使用 Set-IAMRolePermissionsBoundary。

PowerShell V4 的工具

範例 1：此範例示範如何設定 IAM 角色的許可界限。您可以將 AWS 受管政策或自訂政策設定為許可界限。

```
Set-IAMRolePermissionsBoundary -RoleName MyRoleName -PermissionsBoundary  
arn:aws:iam::123456789012:policy/intern-boundary
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [PutRolePermissionsBoundary](#)。

Set-IAMUserPermissionsBoundary

以下程式碼範例顯示如何使用 Set-IAMUserPermissionsBoundary。

PowerShell V4 的工具

範例 1：此範例示範如何設定使用者的許可界限。您可以將 AWS 受管政策或自訂政策設定為許可界限。

```
Set-IAMUserPermissionsBoundary -UserName joe -PermissionsBoundary  
arn:aws:iam::123456789012:policy/intern-boundary
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [PutUserPermissionsBoundary](#)。

Sync-IAMMFADevice

以下程式碼範例顯示如何使用 Sync-IAMMFADevice。

PowerShell V4 的工具

範例 1：此範例會將與 IAM 使用者 **Bob** 關聯且 ARN 為 **arn:aws:iam::123456789012:mfa/bob** 的 MFA 裝置，與提供兩個驗證碼的身分驗證器程式同步。

```
Sync-IAMMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/theresa -  
AuthenticationCode1 123456 -AuthenticationCode2 987654 -UserName Bob
```

範例 2：此範例會將與 IAM 使用者 **Theresa** 關聯的 IAM MFA 裝置，與序號為 **ABCD12345678** 並提供兩個驗證碼的實體裝置同步。

```
Sync-IAMMFADevice -SerialNumber ABCD12345678 -AuthenticationCode1 123456 -  
AuthenticationCode2 987654 -UserName Theresa
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ResyncMfaDevice](#)。

Unregister-IAMGroupPolicy

以下程式碼範例顯示如何使用 Unregister-IAMGroupPolicy。

PowerShell V4 的工具

範例 1：此範例會將 ARN 為 **arn:aws:iam::123456789012:policy/TesterAccessPolicy** 的受管群組政策與名為 **Testers** 的群組分離。

```
Unregister-IAMGroupPolicy -GroupName Testers -PolicyArn  
arn:aws:iam::123456789012:policy/TesterAccessPolicy
```

範例 2：此範例會找出連接到名為 **Testers** 的群組的所有受管政策，並將其與群組分離。

```
Get-IAMAttachedGroupPolicies -GroupName Testers | Unregister-IAMGroupPolicy -  
Groupname Testers
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DetachGroupPolicy](#)。

Unregister-IAMRolePolicy

以下程式碼範例顯示如何使用 Unregister-IAMRolePolicy。

PowerShell V4 的工具

範例 1：此範例會將 ARN 為 **arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy** 的受管群組政策與名為 **FedTesterRole** 的角色分離。

```
Unregister-IAMRolePolicy -RoleName FedTesterRole -PolicyArn  
arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy
```


範例 2：此範例會找出連接到名為 **FedTesterRole** 的角色的所有受管政策，並將其與角色分離。

```
Get-IAMAttachedRolePolicyList -RoleName FedTesterRole | Unregister-IAMRolePolicy -  
Rolename FedTesterRole
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DetachRolePolicy](#)。

Unregister-IAMUserPolicy

以下程式碼範例顯示如何使用 Unregister-IAMUserPolicy。

PowerShell V4 的工具

範例 1：此範例會將 ARN 為 **arn:aws:iam::123456789012:policy/TesterPolicy** 的受管政策與名為 **Bob** 的 IAM 使用者分離。

```
Unregister-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::123456789012:policy/  
TesterPolicy
```

範例 2：此範例會找出連接到名為 **Theresa** 的 IAM 使用者的所有受管政策，並將這些政策與使用者分離。

```
Get-IAMAttachedUserPolicyList -UserName Theresa | Unregister-IAMUserPolicy -Username  
Theresa
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DetachUserPolicy](#)。

Update-IAMAccessKey

以下程式碼範例顯示如何使用 Update-IAMAccessKey。

PowerShell V4 的工具

範例 1：此範例會將名為 **Bob** 之 IAM 使用者的存取金鑰 **AKIAIOSFODNN7EXAMPLE** 的狀態變更為 **Inactive**。

```
Update-IAMAccessKey -UserName Bob -AccessKeyId AKIAIOSFODNN7EXAMPLE -Status Inactive
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UpdateAccessKey](#)。

Update-IAMAccountPasswordPolicy

以下程式碼範例顯示如何使用 Update-IAMAccountPasswordPolicy。

PowerShell V4 的工具

範例 1：此範例會使用指定設定更新帳戶的密碼政策。請注意，命令中未包含的任何參數都不會保持不變。相反，它們會重設為預設值。

```
Update-IAMAccountPasswordPolicy -AllowUsersToChangePasswords $true -HardExpiry  
$false -MaxPasswordAge 90 -MinimumPasswordLength 8 -PasswordReusePrevention 20  
-RequireLowercaseCharacters $true -RequireNumbers $true -RequireSymbols $true -  
RequireUppercaseCharacters $true
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [UpdateAccountPasswordPolicy](#)。

Update-IAMAssumeRolePolicy

以下程式碼範例顯示如何使用 Update-IAMAssumeRolePolicy。

PowerShell V4 的工具

範例 1：此範例會使用新的信任政策更新名為 **ClientRole** 的 IAM 角色，其內容來自 **ClientRolePolicy.json** 檔案。請注意，必須使用 **-Raw** 切換參數，才能成功處理 JSON 檔案的內容。

```
Update-IAMAssumeRolePolicy -RoleName ClientRole -PolicyDocument (Get-Content -raw  
ClientRolePolicy.json)
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [UpdateAssumeRolePolicy](#)。

Update-IAMGroup

以下程式碼範例顯示如何使用 Update-IAMGroup。

PowerShell V4 的工具

範例 1：此範例會將 IAM 群組 **Testers** 重新命名為 **AppTesters**。

```
Update-IAMGroup -GroupName Testers -NewGroupName AppTesters
```

範例 2：此範例會將 IAM 群組 **AppTesters** 的路徑變更為 **/0rg1/0rg2/**。這會將群組的 ARN 變更為 **arn:aws:iam::123456789012:group/0rg1/0rg2/AppTesters**。

```
Update-IAMGroup -GroupName AppTesters -NewPath /0rg1/0rg2/
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [UpdateGroup](#)。

Update-IAMLoginProfile

以下程式碼範例顯示如何使用 Update-IAMLoginProfile。

PowerShell V4 的工具

範例 1：此範例會為 IAM 使用者 **Bob** 設定新的臨時密碼，並在使用者下次登入時要求其變更密碼。

```
Update-IAMLoginProfile -UserName Bob -Password "P@ssw0rd1234" -PasswordResetRequired $true
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [UpdateLoginProfile](#)。

Update-IAMOpenIDConnectProviderThumbprint

以下程式碼範例顯示如何使用 Update-IAMOpenIDConnectProviderThumbprint。

PowerShell V4 的工具

範例 1：此範例會更新 ARN 為 **arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com** 之 OIDC 提供者的憑證指紋清單，以使用新指紋。當與提供者關聯之憑證變更時，OIDC 提供者會共用新值。

```
Update-IAMOpenIDConnectProviderThumbprint -OpenIDConnectProviderArn  
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com -ThumbprintList  
7359755EXAMPLEeabc3060bce3EXAMPLEec4542a3
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [UpdateOpenIdConnectProviderThumbprint](#)。

Update-IAMRole

以下程式碼範例顯示如何使用 Update-IAMRole。

PowerShell V4 的工具

範例 1：此範例會更新角色描述，以及可以請求角色工作階段的工作階段持續時間值上限 (以秒為單位)。

```
Update-IAMRole -RoleName MyRoleName -Description "My testing role" -  
MaxSessionDuration 43200
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UpdateRole](#)。

Update-IAMRoleDescription

以下程式碼範例顯示如何使用 Update-IAMRoleDescription。

PowerShell V4 的工具

範例 1：此範例會更新您的帳戶中 IAM 角色的描述。

```
Update-IAMRoleDescription -RoleName MyRoleName -Description "My testing role"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UpdateRoleDescription](#)。

Update-IAMSAMLProvider

以下程式碼範例顯示如何使用 Update-IAMSAMLProvider。

PowerShell V4 的工具

範例 1：此範例會更新 IAM 中 ARN 為 **arn:aws:iam::123456789012:saml-provider/SAMLADFS**、使用檔案 **SAMLMetaData.xml** 中的新 SAML 中繼資料文件的 SAML 提供者。請注意，必須使用 **-Raw** 切換參數，才能成功處理 JSON 檔案的內容。

```
Update-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS -SAMLMetadataDocument (Get-Content -Raw SAMLMetaData.xml)
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [UpdateSamlProvider](#)。

Update-IAMServerCertificate

以下程式碼範例顯示如何使用 Update-IAMServerCertificate。

PowerShell V4 的工具

範例 1：此範例會將名為 **MyServerCertificate** 的憑證重新命名為 **MyRenamedServerCertificate**。

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -
NewServerCertificateName MyRenamedServerCertificate
```

範例 2：此範例會將名為 **MyServerCertificate** 的憑證移至路徑 **/Org1/Org2/**。這會將資源的 ARN 變更為 **arn:aws:iam::123456789012:server-certificate/Org1/Org2/MyServerCertificate**。

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -NewPath /
Org1/Org2/
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [UpdateServerCertificate](#)。

Update-IAMSigningCertificate

以下程式碼範例顯示如何使用 Update-IAMSigningCertificate。

PowerShell V4 的工具

範例 1：此範例會更新與名為 **Bob**、憑證 ID 為 **Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU** 的 IAM 使用者關聯的憑證，將其標示為非作用中。

```
Update-IAMSigningCertificate -CertificateId Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU -
  UserName Bob -Status Inactive
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [UpdateSigningCertificate](#)。

Update-IAMUser

以下程式碼範例顯示如何使用 Update-IAMUser。

PowerShell V4 的工具

範例 1：此範例會將 IAM 使用者 **Bob** 重新命名為 **Robert**。

```
Update-IAMUser -UserName Bob -NewUserName Robert
```

範例 2：此範例會將 IAM 使用者 **Bob** 的路徑變更為 **/0rg1/0rg2/**，這可有效地將使用者的 ARN 變更為 **arn:aws:iam::123456789012:user/0rg1/0rg2/bob**。

```
Update-IAMUser -UserName Bob -NewPath /0rg1/0rg2/
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [UpdateUser](#)。

Write-IAMGroupPolicy

以下程式碼範例顯示如何使用 Write-IAMGroupPolicy。

PowerShell V4 的工具

範例 1：此範例會建立名為 **AppTesterPolicy** 的內嵌政策，並將其內嵌在 IAM 群組 **AppTesters** 中。如果已存在同名的內嵌政策，其會被覆寫。JSON 政策內容來自 **apptesterpolicy.json** 檔案。請注意，必須使用 **-Raw** 參數，才能成功處理 JSON 檔案的內容。

```
Write-IAMGroupPolicy -GroupName AppTesters -PolicyName AppTesterPolicy -  
PolicyDocument (Get-Content -Raw apptesterpolicy.json)
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [PutGroupPolicy](#)。

Write-IAMRolePolicy

以下程式碼範例顯示如何使用 Write-IAMRolePolicy。

PowerShell V4 的工具

範例 1：此範例會建立名為 **FedTesterRolePolicy** 的內嵌政策，並將其內嵌在 IAM 角色 **FedTesterRole** 中。如果已存在同名的內嵌政策，其會被覆寫。JSON 政策內容來自 **FedTesterPolicy.json** 檔案。請注意，必須使用 **-Raw** 參數，才能成功處理 JSON 檔案的內容。

```
Write-IAMRolePolicy -RoleName FedTesterRole -PolicyName FedTesterRolePolicy -  
PolicyDocument (Get-Content -Raw FedTesterPolicy.json)
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [PutRolePolicy](#)。

Write-IAMUserPolicy

以下程式碼範例顯示如何使用 Write-IAMUserPolicy。

PowerShell V4 的工具

範例 1：此範例會建立名為 **EC2AccessPolicy** 的內嵌政策，並將其內嵌在 IAM 使用者 **Bob** 中。如果已存在同名的內嵌政策，其會被覆寫。JSON 政策內容來自 **EC2AccessPolicy.json** 檔案。請注意，必須使用 **-Raw** 參數，才能成功處理 JSON 檔案的內容。

```
Write-IAMUserPolicy -UserName Bob -PolicyName EC2AccessPolicy -PolicyDocument (Get-  
Content -Raw EC2AccessPolicy.json)
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [PutUserPolicy](#)。

使用 Tools for PowerShell V4 的 Kinesis 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 Kinesis 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Get-KINRecord

以下程式碼範例顯示如何使用 Get-KINRecord。

PowerShell V4 的工具

範例 1：此範例示範如何從一系列的一或多個記錄傳回和擷取資料。提供給 Get-KINRecord 的迭代器會決定記錄的開始位置，以傳回在此範例中擷取到變數 \$records 的位置。然後，您可以透過編製 \$records 集合的索引來存取每個個別記錄。假設記錄中的資料為 UTF-8 編碼文字，最終命令會示範如何從物件中的 MemoryStream 擷取資料，並將其做為文字傳回主控台。

```
$records
$records = Get-KINRecord -ShardIterator "AAAAAAAAAAGIc....9VnbiRNaP"
```

輸出：

MillisBehindLatest	NextShardIterator	Records
0	AAAAAAAAAAERNIq...uDn11HuUs	{Key1, Key2}

```
$records.Records[0]
```

輸出：

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())
```

輸出：

```
test data from string
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GetRecords](#)。

Get-KINShardIterator

以下程式碼範例顯示如何使用 Get-KINShardIterator。

PowerShell V4 的工具

範例 1：傳回指定碎片和開始位置的碎片疊代運算。碎片識別符和序號的詳細資訊可從 Get-KINStream cmdlet 的輸出取得，方法是參考傳回串流物件的碎片集合。傳回的迭代器可與 Get-KINRecord cmdlet 搭配使用，以提取碎片中的資料記錄。

```
Get-KINShardIterator -StreamName "mystream" -ShardId "shardId-000000000000" -  
ShardIteratorType AT_SEQUENCE_NUMBER -StartingSequenceNumber "495598645..."
```

輸出：

```
AAAAAAAAAAGIc....9VnbiRNaP
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [GetShardIterator](#)。

Get-KINStream

以下程式碼範例顯示如何使用 Get-KINStream。

PowerShell V4 的工具

範例 1：傳回指定串流的詳細資訊。

```
Get-KINStream -StreamName "mystream"
```

輸出：

```
HasMoreShards      : False
RetentionPeriodHours : 24
Shards              : {}
StreamARN           : arn:aws:kinesis:us-west-2:123456789012:stream/mystream
StreamName          : mystream
StreamStatus        : ACTIVE
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeStream](#)。

New-KINStream

以下程式碼範例顯示如何使用 New-KINStream。

PowerShell V4 的工具

範例 1：建立新的串流。根據預設，此 cmdlet 不會傳回任何輸出，因此會新增 -PassThru 開關，以傳回提供給 -StreamName 參數的值，以供後續使用。

```
$streamName = New-KINStream -StreamName "mystream" -ShardCount 1 -PassThru
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateStream](#)。

Remove-KINStream

以下程式碼範例顯示如何使用 Remove-KINStream。

PowerShell V4 的工具

範例 1：刪除指定的串流。在命令執行之前，系統會提示您進行確認。若要隱藏確認提示，請使用 -Force 切換。

```
Remove-KINStream -StreamName "mystream"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteStream](#)。

Write-KINRecord

以下程式碼範例顯示如何使用 Write-KINRecord。

PowerShell V4 的工具

範例 1：撰寫包含提供給 -Text 參數之字串的記錄。

```
Write-KINRecord -Text "test data from string" -StreamName "mystream" -PartitionKey "Key1"
```

範例 2：寫入包含指定檔案中資料的記錄。該檔案會被視為位元組序列，因此如果其包含文字，則應在搭配此 cmdlet 使用它之前，以任何必要的編碼撰寫。

```
Write-KINRecord -FilePath "C:\TestData.txt" -StreamName "mystream" -PartitionKey "Key2"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [PutRecord](#)。

使用 Tools for PowerShell V4 的 Lambda 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 Lambda 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Add-LMResourceTag

以下程式碼範例顯示如何使用 Add-LMResourceTag。

PowerShell V4 的工具

範例 1：將三個標籤 (華盛頓、奧勒岡和加利佛尼亞) 及其相關聯的值新增至其 ARN 識別的指定函數。

```
Add-LMResourceTag -Resource "arn:aws:lambda:us-west-2:123456789012:function:MyFunction" -Tag @{ "Washington" = "Olympia"; "Oregon" = "Salem"; "California" = "Sacramento" }
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [TagResource](#)。

Get-LMAccountSetting

以下程式碼範例顯示如何使用 Get-LMAccountSetting。

PowerShell V4 的工具

範例 1：此範例會顯示以比較帳戶限制和帳戶用量

```
Get-LMAccountSetting | Select-Object  
@{Name="TotalCodeSizeLimit";Expression={$_.AccountLimit.TotalCodeSize}},  
@{Name="TotalCodeSizeUsed";Expression={$_.AccountUsage.TotalCodeSize}}
```

輸出：

TotalCodeSizeLimit	TotalCodeSizeUsed
80530636800	15078795

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetAccountSettings](#)。

Get-LMAlias

以下程式碼範例顯示如何使用 Get-LMAlias。

PowerShell V4 的工具

範例 1：此範例會擷取特定 Lambda 函數別名的路由組態權重。

```
Get-LMAlias -FunctionName "MyLambdaFunction123" -Name "newlabel1" -Select  
RoutingConfig
```

輸出：

```
AdditionalVersionWeights  
-----  
{[1, 0.6]}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GetAlias](#)。

Get-LMFunctionConcurrency

以下程式碼範例顯示如何使用 Get-LMFunctionConcurrency。

PowerShell V4 的工具

範例 1：此範例會取得 Lambda 函數的預留並行

```
Get-LMFunctionConcurrency -FunctionName "MyLambdaFunction123" -Select *
```

輸出：

```
ReservedConcurrentExecutions  
-----  
100
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetFunctionConcurrency](#)。

Get-LMFunctionConfiguration

以下程式碼範例顯示如何使用 Get-LMFunctionConfiguration。

PowerShell V4 的工具

範例 1：此範例會傳回 Lambda 函數的版本特定組態。

```
Get-LMFunctionConfiguration -FunctionName "MylambdaFunction123" -Qualifier
"PowershellAlias"
```

輸出：

```
CodeSha256           : uW0W0R7z+f0VyLuUg7+/D08hkMFsq0SF4seuyUZJ/R8=
CodeSize             : 1426
DeadLetterConfig     : Amazon.Lambda.Model.DeadLetterConfig
Description           : Version 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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetFunctionConfiguration](#)。

Get-LMFunctionList

以下程式碼範例顯示如何使用 Get-LMFunctionList。

PowerShell V4 的工具

範例 1：此範例會顯示具有已排序程式碼大小的所有 Lambda 函數

```
Get-LMFunctionList | Sort-Object -Property CodeSize | Select-Object FunctionName,
RunTime, Timeout, CodeSize
```

輸出：

FunctionName CodeSize ----- -----	Runtime	Timeout
test 243	python2.7	3
MylambdaFunction123 659	python3.8	600
myfuncpython1 675	python3.8	303

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListFunctions](#)。

Get-LMPolicy

以下程式碼範例顯示如何使用 Get-LMPolicy。

PowerShell V4 的工具

範例 1：此範例顯示 Lambda 函數的函數政策

```
Get-LMPolicy -FunctionName test -Select Policy
```

輸出：

```
{"Version":"2012-10-17","Id":"default","Statement":
[{"Sid":"xxxx","Effect":"Allow","Principal":
```

```
{"Service":"sns.amazonaws.com"},"Action":"lambda:InvokeFunction","Resource":"arn:aws:lambda:us-east-1:123456789102:function:test"]}]}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GetPolicy](#)。

Get-LMProvisionedConcurrencyConfig

以下程式碼範例顯示如何使用 Get-LMProvisionedConcurrencyConfig。

PowerShell V4 的工具

範例 1：此範例會取得 Lambda 函數指定別名的已佈建並行組態。

```
C:\>Get-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -
Qualifier "NewAlias1"
```

輸出：

```
AllocatedProvisionedConcurrentExecutions : 0
AvailableProvisionedConcurrentExecutions : 0
LastModified                             : 2020-01-15T03:21:26+0000
RequestedProvisionedConcurrentExecutions : 70
Status                                    : IN_PROGRESS
StatusReason                             :
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetProvisionedConcurrencyConfig](#)。

Get-LMProvisionedConcurrencyConfigList

以下程式碼範例顯示如何使用 Get-LMProvisionedConcurrencyConfigList。

PowerShell V4 的工具

範例 1：此範例會擷取 Lambda 函數的已佈建並行組態清單。

```
Get-LMProvisionedConcurrencyConfigList -FunctionName "MylambdaFunction123"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListProvisionedConcurrencyConfigs](#)。

Get-LMResourceTag

以下程式碼範例顯示如何使用 Get-LMResourceTag。

PowerShell V4 的工具

範例 1：擷取目前在指定函數中設定的標籤及其值。

```
Get-LMResourceTag -Resource "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
```

輸出：

Key	Value
---	-----
California	Sacramento
Oregon	Salem
Washington	Olympia

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListTags](#)。

Get-LMVersionsByFunction

以下程式碼範例顯示如何使用 Get-LMVersionsByFunction。

PowerShell V4 的工具

範例 1：此範例會傳回每個 Lambda 函數版本的版本特定組態清單。

```
Get-LMVersionsByFunction -FunctionName "MylambdaFunction123"
```

輸出：

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

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [ListVersionsByFunction](#)。

New-LMAlias

以下程式碼範例顯示如何使用 New-LMAlias。

PowerShell V4 的工具

範例 1：此範例會為指定的版本和路由組態建立新的 Lambda 別名，以指定其收到的調用請求百分比。

```

New-LMAlias -FunctionName "MylambdaFunction123" -
RoutingConfig_AdditionalVersionWeight @{Name="1";Value="0.6"} -Description "Alias for
version 4" -FunctionVersion 4 -Name "PowershellAlias"

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateAlias](#)。

Publish-LMFunction

以下程式碼範例顯示如何使用 Publish-LMFunction。

PowerShell V4 的工具

範例 1：此範例會在 AWS Lambda 中建立新的 C# (dotnetcore1.0 執行時間) 函數名為 MyFunction，提供從本機檔案系統上的 zip 檔案（可使用相對或絕對路徑）編譯的函數二進位檔。C# Lambda 函數會使用 AssemblyName::Namespace.ClassName::MethodName 指派方法來指定函數的處理常式。您應該適當地取代處理常式規格的組件名稱 (不含 .dll 尾碼)、命名空間、類別名稱和方法名稱部分。新函數將透過提供的值來設定環境變數 'envvar1' 和 'envvar2'。

```

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" }
```

輸出：

```
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        :
```

範例 2：此範例類似於上一個範例，但函數二進位檔會先上傳到 Amazon S3 儲存貯體 (必須與預期的 Lambda 函數位於相同區域)，然後在建立函數時參考產生的 S3 物件。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key MyFunctionBinaries.zip -File .
\MyFunctionBinaries.zip
Publish-LMFunction -Description "My C# Lambda Function" `
    -FunctionName MyFunction `
    -BucketName amzn-s3-demo-bucket `
    -Key MyFunctionBinaries.zip `
    -Handler "AssemblyName::Namespace.ClassName::MethodName" `
    -Role "arn:aws:iam::123456789012:role/LambdaFullExecRole" `
    -Runtime dotnetcore1.0 `
    -Environment_Variable @{ "envvar1"="value";"envvar2"="value" }
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateFunction](#)。

Publish-LMVersion

以下程式碼範例顯示如何使用 Publish-LMVersion。

PowerShell V4 的工具

範例 1：此範例會為 Lambda 函數程式碼的現有快照建立版本

```
Publish-LMVersion -FunctionName "MyLambdaFunction123" -Description "Publishing Existing Snapshot of function code as a new version through Powershell"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [PublishVersion](#)。

Remove-LMAlias

以下程式碼範例顯示如何使用 Remove-LMAlias。

PowerShell V4 的工具

範例 1：此範例會刪除命令中提到的 Lambda 函數別名。

```
Remove-LMAlias -FunctionName "MyLambdaFunction123" -Name "NewAlias"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteAlias](#)。

Remove-LMFunction

以下程式碼範例顯示如何使用 Remove-LMFunction。

PowerShell V4 的工具

範例 1：此範例會刪除 Lambda 函數的特定版本

```
Remove-LMFunction -FunctionName "MyLambdaFunction123" -Qualifier '3'
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteFunction](#)。

Remove-LMFunctionConcurrency

以下程式碼範例顯示如何使用 Remove-LMFunctionConcurrency。

PowerShell V4 的工具

範例 1：此範例會移除 Lambda 函數的函數並行。

```
Remove-LMFunctionConcurrency -FunctionName "MylambdaFunction123"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteFunctionConcurrency](#)。

Remove-LMPermission

以下程式碼範例顯示如何使用 Remove-LMPermission。

PowerShell V4 的工具

範例 1：此範例會移除 Lambda 函數之指定 StatementId 的函數政策。

```
$policy = Get-LMPolicy -FunctionName "MylambdaFunction123" -Select Policy |  
ConvertFrom-Json | Select-Object -ExpandProperty Statement  
Remove-LMPermission -FunctionName "MylambdaFunction123" -StatementId $policy[0].Sid
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [RemovePermission](#)。

Remove-LMProvisionedConcurrencyConfig

以下程式碼範例顯示如何使用 Remove-LMProvisionedConcurrencyConfig。

PowerShell V4 的工具

範例 1：此範例會移除特定別名的已佈建並行組態。

```
Remove-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -Qualifier  
"NewAlias1"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet 參考》(V4) 中的 [DeleteProvisionedConcurrencyConfig](#)。

Remove-LMResourceTag

以下程式碼範例顯示如何使用 Remove-LMResourceTag。

PowerShell V4 的工具

範例 1：從函數中移除提供的標籤。除非已指定 -Force 切換，否則 cmdlet 會提示進行確認。對服務進行單一呼叫以移除標籤。

```
Remove-LMResourceTag -Resource "arn:aws:lambda:us-west-2:123456789012:function:MyFunction" -TagKey "Washington","Oregon","California"
```

範例 2：從函數中移除提供的標籤。除非已指定 -Force 切換，否則 cmdlet 會提示進行確認。按照提供的標籤對服務進行呼叫後。

```
"Washington","Oregon","California" | Remove-LMResourceTag -Resource "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [UntagResource](#)。

Update-LMAlias

以下程式碼範例顯示如何使用 Update-LMAlias。

PowerShell V4 的工具

範例 1：此範例會更新現有 Lambda 函數別名的組態。它會更新 RoutingConfiguration 值，將 60% (0.6) 的流量轉移到第 1 版

```
Update-LMAlias -FunctionName "MylambdaFunction123" -Description "Alias for version 2" -FunctionVersion 2 -Name "newlabel1" -RoutingConfig_AdditionalVersionWeight @{Name="1";Value="0.6"}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UpdateAlias](#)。

Update-LMFunctionCode

以下程式碼範例顯示如何使用 Update-LMFunctionCode。

PowerShell V4 的工具

範例 1：使用指定 zip 檔案中包含的新內容，更新名為 'MyFunction' 的函數。對於 C# .NET Core Lambda 函數，zip 檔案應包含已編譯的組件。

```
Update-LMFunctionCode -FunctionName MyFunction -ZipFilename .\UpdatedCode.zip
```

範例 2：此範例類似於上一個範例，但使用包含更新程式碼的 Amazon S3 物件來更新函數。

```
Update-LMFunctionCode -FunctionName MyFunction -BucketName amzn-s3-demo-bucket -Key UpdatedCode.zip
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UpdateFunctionCode](#)。

Update-LMFunctionConfiguration

以下程式碼範例顯示如何使用 Update-LMFunctionConfiguration。

PowerShell V4 的工具

範例 1：此範例會更新現有的 Lambda 函數組態

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [UpdateFunctionConfiguration](#)。

Write-LMFunctionConcurrency

以下程式碼範例顯示如何使用 Write-LMFunctionConcurrency。

PowerShell V4 的工具

範例 1：此範例會整體套用函數的並行設定。

```
Write-LMFunctionConcurrency -FunctionName "MyLambdaFunction123" -  
ReservedConcurrentExecution 100
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [PutFunctionConcurrency](#)。

Write-LMProvisionedConcurrencyConfig

以下程式碼範例顯示如何使用 Write-LMProvisionedConcurrencyConfig。

PowerShell V4 的工具

範例 1：此範例會將佈建並行組態新增至函數別名

```
Write-LMProvisionedConcurrencyConfig -FunctionName "MyLambdaFunction123" -  
ProvisionedConcurrentExecution 20 -Qualifier "NewAlias1"
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [PutProvisionedConcurrencyConfig](#)。

使用 Tools for PowerShell V4 的 Amazon ML 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 Amazon ML 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Get-MLBatchPrediction

以下程式碼範例顯示如何使用 Get-MLBatchPrediction。

PowerShell V4 的工具

範例 1：傳回具有 ID 之批次預測的詳細中繼資料。

```
Get-MLBatchPrediction -BatchPredictionId ID
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetBatchPrediction](#)。

Get-MLBatchPredictionList

以下程式碼範例顯示如何使用 Get-MLBatchPredictionList。

PowerShell V4 的工具

範例 1：傳回所有 BatchPredictions 及其相關資料記錄的清單，這些記錄符合請求中提供的搜尋條件。

```
Get-MLBatchPredictionList
```

範例 2：傳回狀態為 COMPLETED 的所有 BatchPredictions 清單。

```
Get-MLBatchPredictionList -FilterVariable Status -EQ COMPLETED
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeBatchPredictions](#)。

Get-MLDataSource

以下程式碼範例顯示如何使用 Get-MLDataSource。

PowerShell V4 的工具

範例 1：傳回 ID 為之 DataSource 的中繼資料、狀態和資料檔案資訊

```
Get-MLDataSource -DataSourceId ID
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [GetDataSource](#)。

Get-MLDataSourceList

以下程式碼範例顯示如何使用 Get-MLDataSourceList。

PowerShell V4 的工具

範例 1：傳回所有 DataSources 及其相關聯資料記錄的清單。

```
Get-MLDataSourceList
```

範例 2：傳回狀態為 COMPLETED 的所有 DataSources 清單。

```
Get-MLDataDourceList -FilterVariable Status -EQ COMPLETED
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeDataSources](#)。

Get-MLEvaluation

以下程式碼範例顯示如何使用 Get-MLEvaluation。

PowerShell V4 的工具

範例 1：傳回 ID 為 之評估的中繼資料和狀態。

```
Get-MLEvaluation -EvaluationId ID
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetEvaluation](#)。

Get-MLEvaluationList

以下程式碼範例顯示如何使用 Get-MLEvaluationList。

PowerShell V4 的工具

範例 1：傳回所有評估資源的清單

```
Get-MLEvaluationList
```

範例 2：傳回狀態為 COMPLETED 的所有疏散清單。

```
Get-MLEvaluationList -FilterVariable Status -EQ COMPLETED
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeEvaluations](#)。

Get-MLModel

以下程式碼範例顯示如何使用 Get-MLModel。

PowerShell V4 的工具

範例 1：傳回 ID 為 之 MLModel 的詳細資訊中繼資料、狀態、結構描述和資料檔案資訊。

```
Get-MLModel -ModelId ID
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [GetMLModel](#)。

Get-MLModelList

以下程式碼範例顯示如何使用 Get-MLModelList。

PowerShell V4 的工具

範例 1：傳回所有模型及其相關聯資料記錄的清單。

```
Get-MLModelList
```

範例 2：傳回狀態為 COMPLETED 的所有模型清單。

```
Get-MLModelList -FilterVariable Status -EQ COMPLETED
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeMLModels](#)。

Get-MLPrediction

以下程式碼範例顯示如何使用 Get-MLPrediction。

PowerShell V4 的工具

範例 1：將記錄傳送至 ID 為 之模型的即時預測端點 URL。

```
Get-MLPrediction -ModelId ID -PredictEndpoint URL -Record @"A" = "B"; "C" = "D";}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的[預測](#)。

New-MLBatchPrediction

以下程式碼範例顯示如何使用 New-MLBatchPrediction。

PowerShell V4 的工具

範例 1：為 ID 為 的模型建立新的批次預測請求，並將輸出放在指定的 S3 位置。

```
New-MLBatchPrediction -ModelId ID -Name NAME -OutputURI s3://...
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的[CreateBatchPrediction](#)。

New-MLDataSourceFromS3

以下程式碼範例顯示如何使用 New-MLDataSourceFromS3。

PowerShell V4 的工具

範例 1：使用 S3 位置的資料建立資料來源，名稱為 NAME，結構描述為 SCHEMA。

```
New-MLDataSourceFromS3 -Name NAME -ComputeStatistics $true -DataSpec_DataLocationS3  
"s3://BUCKET/KEY" -DataSchema SCHEMA
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的[CreateDataSourceFromS3](#)。

New-MLEvaluation

以下程式碼範例顯示如何使用 New-MLEvaluation。

PowerShell V4 的工具

範例 1：為指定的資料來源 ID 和模型 ID 建立評估

```
New-MLEvaluation -Name NAME -DataSourceId DSID -ModelId MID
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateEvaluation](#)。

New-MLModel

以下程式碼範例顯示如何使用 New-MLModel。

PowerShell V4 的工具

範例 1：使用訓練資料建立新的模型。

```
New-MLModel -Name NAME -ModelType BINARY -Parameter @{...} -TrainingDataSourceId ID
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateMLModel](#)。

New-MLRealtimeEndpoint

以下程式碼範例顯示如何使用 New-MLRealtimeEndpoint。

PowerShell V4 的工具

範例 1：為指定的模型 ID 建立新的即時預測端點。

```
New-MLRealtimeEndpoint -ModelId ID
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateRealtimeEndpoint](#)。

使用 Tools for PowerShell V4 的 Macie 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 Macie 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Get-MAC2FindingList

以下程式碼範例顯示如何使用 Get-MAC2FindingList。

PowerShell V4 的工具

範例 1：傳回包含類型為 "CREDIT_CARD_NUMBER" 或 "US_SOCIAL_SECURITY_NUMBER" 之敏感資料偵測之調查結果的 FindingIds 清單

```
$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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListFindings](#)。

AWS OpsWorks 使用 Tools for PowerShell V4 的範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 來執行動作和實作常見案例 AWS OpsWorks。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

New-OPSDeployment

以下程式碼範例顯示如何使用 New-OPSDeployment。

PowerShell V4 的工具

範例 1：此命令會在 layer in AWS OpsWorks Stacks 中的所有 Linux 型執行個體上建立新的應用程式部署。即使您指定 layer ID，也必須指定堆疊 ID。命令可讓部署在需要時重新啟動執行個體。

```
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"}
```

範例 2：此命令會從 **phpapp** 技術指南部署 **appsetup** 配方，以及從 **testcookbook** 技術指南部署 **secbaseline** 配方。部署目標是一個執行個體，但也需要堆疊 ID 和 layer ID。Command_Arg 參數 **allow_reboot** 屬性設定為 **true**，讓部署在需要時重新啟動執行個體。

```
$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"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateDeployment](#)。

AWS 價格表 使用 Tools for PowerShell V4 的範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 來執行動作和實作常見案例 AWS 價格表。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Get-PLSAttributeValue

以下程式碼範例顯示如何使用 Get-PLSAttributeValue。

PowerShell V4 的工具

範例 1：傳回 us-east-1 區域中 Amazon EC2 屬性 'volumeType' 的值。

```
Get-PLSAttributeValue -ServiceCode AmazonEC2 -AttributeName "volumeType" -region us-east-1
```

輸出：

```
Value
-----
Cold HDD
General Purpose
Magnetic
Provisioned IOPS
Throughput Optimized HDD
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetAttributeValues](#)。

Get-PLSProduct

以下程式碼範例顯示如何使用 Get-PLSProduct。

PowerShell V4 的工具

範例 1：傳回 Amazon EC2 所有產品的詳細資訊。

```
Get-PLSProduct -ServiceCode AmazonEC2 -Region us-east-1
```

輸出：

```
{"product":{"productFamily":"Compute Instance","attributes":{"enhancedNetworkingSupported":"Yes","memory":"30.5 GiB","dedicatedEbsThroughput":"8000 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"},...}}
```

範例 2：傳回由「一般用途」磁碟區類型篩選的 us-east-1 區域中的 Amazon EC2 資料，這些磁碟區類型為 SSD 後端。

```
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
```

輸出：

```
{"product":{"productFamily":"Storage","attributes":{"storageMedia":"SSD-backed","maxThroughputvolume":"160 MB/sec","volumeType":"General Purpose","maxIopsvolume":"10000"},...}}
```

- 如需 API 詳細資訊，請參閱《Cmdlet 參考》中的 [GetProducts](#) (V4)。AWS Tools for PowerShell

Get-PLSService

以下程式碼範例顯示如何使用 Get-PLSService。

PowerShell V4 的工具

範例 1：傳回 us-east-1 區域中所有可用服務代碼的中繼資料。

```
Get-PLSService -Region us-east-1
```

輸出：

AttributeNames	ServiceCode
-----	-----
{productFamily, servicecode, groupDescription, termType...}	AWSBudgets
{productFamily, servicecode, termType, usagetype...}	AWSCloudTrail
{productFamily, servicecode, termType, usagetype...}	AWSCodeCommit
{productFamily, servicecode, termType, usagetype...}	AWSCodeDeploy
{productFamily, servicecode, termType, usagetype...}	AWSCodePipeline
{productFamily, servicecode, termType, usagetype...}	AWSConfig
...	

範例 2：傳回 us-east-1 區域中 Amazon EC2 服務的中繼資料。

```
Get-PLSService -ServiceCode AmazonEC2 -Region us-east-1
```

輸出：

AttributeNames	ServiceCode
-----	-----
{volumeType, maxIopsvolume, instanceCapacity10xlarge, locationType...}	AmazonEC2

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeServices](#)。

使用 Tools for PowerShell V4 的資源群組範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配資源群組來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執程式碼的指示。

主題

- [動作](#)

動作

Add-RGResourceTag

以下程式碼範例顯示如何使用 Add-RGResourceTag。

PowerShell V4 的工具

範例 1：此範例會將值為 'workboxes' 的標籤索引鍵 'Instances' 新增至指定的資源群組 arn

```
Add-RGResourceTag -Tag @{Instances="workboxes"} -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes
```

輸出：

Arn	Tags
---	----
arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes	{[Instances, workboxes]}

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [標籤](#)。

Find-RGResource

以下程式碼範例顯示如何使用 Find-RGResource。

PowerShell V4 的工具

範例 1：此範例會使用標籤篩選條件為執行個體資源類型建立 ResourceQuery，並尋找資源。

```
$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
```

輸出：

ResourceArn	ResourceType
-----	-----
arn:aws:ec2:eu-west-1:123456789012:instance/i-0123445b6cb7bd67b	AWS::EC2::Instance

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [SearchResources](#)。

Get-RGGroup

以下程式碼範例顯示如何使用 Get-RGGroup。

PowerShell V4 的工具

範例 1：此範例會根據群組名稱擷取資源群組

```
Get-RGGroup -GroupName auto-no
```

輸出：

Description	GroupArn	Name
-----	-----	----
	arn:aws:resource-groups:eu-west-1:123456789012:group/auto-no	auto-no

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetGroup](#)。

Get-RGGroupList

以下程式碼範例顯示如何使用 Get-RGGroupList。

PowerShell V4 的工具

範例 1：此範例列出已建立的資源群組。

```
Get-RGGroupList
```

輸出：

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListGroups](#)。

Get-RGGroupQuery

以下程式碼範例顯示如何使用 Get-RGGroupQuery。

PowerShell V4 的工具

範例 1：此範例會擷取指定資源群組的資源查詢

```
Get-RGGroupQuery -GroupName auto-no | Select-Object -ExpandProperty ResourceQuery
```

輸出：

Query	Type
-----	----
<pre>{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":[{"Key":"auto","Values":["no"]}]} TAG_FILTERS_1_0</pre>	

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetGroupQuery](#)。

Get-RGGroupResourceList

以下程式碼範例顯示如何使用 Get-RGGroupResourceList。

PowerShell V4 的工具

範例 1：此範例根據依資源類型篩選來列出群組資源

```
Get-RGGroupResourceList -Filter @{Name="resource-type";Values="AWS::EC2::Instance"}
-GroupName auto-yes | Select-Object -ExpandProperty ResourceIdentifiers
```

輸出：

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListGroupResources](#)。

Get-RGResourceTag

以下程式碼範例顯示如何使用 Get-RGResourceTag。

PowerShell V4 的工具

範例 1：此範例列出指定資源群組 arn 的標籤

```
Get-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes
```

輸出：

Key	Value
---	----
Instances	workboxes

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GetTags](#)。

New-RGGroup

以下程式碼範例顯示如何使用 New-RGGroup。

PowerShell V4 的工具

範例 1：此範例會建立新的標籤型 AWS 資源群組資源群組，名為 TestPowerShellGroup。群組包含目前區域中以標籤鍵 "Name" 和標籤值 "test2" 標記的 Amazon EC2 執行個體。命令會傳回 群組的查詢和類型，以及 操作的結果。

```
$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."
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateGroup](#)。

Remove-RGGroup

以下程式碼範例顯示如何使用 Remove-RGGroup。

PowerShell V4 的工具

範例 1：此範例會移除具名資源群組

```
Remove-RGGroup -GroupName non-tag-cfn-elbv2
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteGroup](#)。

Remove-RGResourceTag

以下程式碼範例顯示如何使用 Remove-RGResourceTag。

PowerShell V4 的工具

範例 1：此範例會從資源群組中移除提及的標籤

```
Remove-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/
workboxes -Key Instances
```

輸出：

```
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}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的[取消標記](#)。

Update-RGGroup

以下程式碼範例顯示如何使用 Update-RGGroup。

PowerShell V4 的工具

範例 1：此範例會更新群組的描述

```
Update-RGGroup -GroupName auto-yes -Description "Instances auto-remove"
```

輸出：

```
Description          GroupArn
Name
-----
----
Instances to be cleaned arn:aws:resource-groups:eu-west-1:123456789012:group/auto-
yes auto-yes
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的[UpdateGroup](#)。

Update-RGGroupQuery

以下程式碼範例顯示如何使用 Update-RGGroupQuery。

PowerShell V4 的工具

範例 1：此範例會建立查詢物件，並更新 群組的查詢。

```
$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
```

輸出：

```
GroupName ResourceQuery
-----
build600   Amazon.ResourceGroups.Model.ResourceQuery
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UpdateGroupQuery](#)。

使用 Tools for PowerShell V4 標記 API 範例的資源群組

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配資源群組標記 API 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執程式碼的指示。

主題

- [動作](#)

動作

Add-RGResourceTag

以下程式碼範例顯示如何使用 Add-RGResourceTag。

PowerShell V4 的工具

範例 1：此範例會將值為 "beta" 和 "preprod_test" 的標籤金鑰 "stage" 和 "version" 新增至 Amazon S3 儲存貯體和 Amazon DynamoDB 資料表。對服務進行單一呼叫以套用標籤。

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

Add-RGTResourceTag -ResourceARNList $arn1,$arn2 -Tag @{ "stage"="beta";
"version"="preprod_test" }
```

範例 2：此範例會將指定的標籤和值新增至 Amazon S3 儲存貯體和 Amazon DynamoDB 資料表。對服務進行兩次呼叫，每個資源 ARN 一個呼叫會輸送到 cmdlet。

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

$arn1,$arn2 | Add-RGTResourceTag -Tag @{ "stage"="beta"; "version"="preprod_test" }
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [TagResources](#)。

Get-RGTResource

以下程式碼範例顯示如何使用 Get-RGTResource。

PowerShell V4 的工具

範例 1：傳回區域中所有已標記的資源，以及與資源相關聯的標籤索引鍵。如果沒有將 -Region 參數提供給 cmdlet，則會嘗試從 shell 或 EC2 執行個體中繼資料推斷區域。

```
Get-RGTResource
```

輸出：

ResourceARN	Tags
-----	----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable	{stage, version}
arn:aws:s3:::amzn-s3-demo-bucket	{stage,
version, othertag}	

範例 2：傳回區域中指定類型的所有已標記資源。每個服務名稱和資源類型的字串與內嵌在資源的 Amazon Resource Name (ARN) 中的字串相同。

```
Get-RGTResource -ResourceType "s3"
```

輸出：

ResourceARN	Tags
-----	----
arn:aws:s3:::amzn-s3-demo-bucket	{stage,
version, othertag}	

範例 3：傳回區域中指定類型的所有已標記資源。請注意，當資源類型輸送至 cmdlet 時，會針對每個提供的資源類型對服務進行一次呼叫。

```
"dynamodb","s3" | Get-RGTResource
```

輸出：

ResourceARN	Tags
-----	----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable	{stage, version}
arn:aws:s3:::amzn-s3-demo-bucket	{stage,
version, othertag}	

範例 4：傳回符合指定篩選條件的所有已標記資源。

```
Get-RGTResource -TagFilter @{ Key="stage" }
```

輸出：

ResourceARN	Tags
-----	----
arn:aws:s3:::amzn-s3-demo-bucket	{stage,
version, othertag}	

範例 5：傳回符合指定篩選條件和資源類型的所有已標記資源。

```
Get-RGTResource -TagFilter @{ Key="stage" } -ResourceType "dynamodb"
```

輸出：

ResourceARN	Tags
-----	----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable	{stage, version}

範例 6：傳回符合指定篩選條件的所有已標記資源。

```
Get-RGTResource -TagFilter @{ Key="stage"; Values=@("beta","gamma") }
```

輸出：

ResourceARN	Tags
-----	----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable	{stage, version}

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetResources](#)。

Get-RGTTagKey

以下程式碼範例顯示如何使用 Get-RGTTagKey。

PowerShell V4 的工具

範例 1：傳回指定區域中的所有標籤索引鍵。如果未指定 -Region 參數，Cmdlet 會嘗試從預設 Shell 區域或 EC2 執行個體中繼資料推斷區域。請注意，標籤索引鍵不會以任何特定順序傳回。

```
Get-RGTTagKey -region us-west-2
```

輸出：

```
version
stage
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetTagKeys](#)。

Get-RGTTagValue

以下程式碼範例顯示如何使用 Get-RGTTagValue。

PowerShell V4 的工具

範例 1：傳回區域中指定標籤的值。如果未指定 -Region 參數，Cmdlet 會嘗試從預設 Shell 區域或 EC2 執行個體中繼資料推斷區域。

```
Get-RGTTagValue -Key "stage" -Region us-west-2
```

輸出：

```
beta
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetTagValues](#)。

Remove-RGTResourceTag

以下程式碼範例顯示如何使用 Remove-RGTResourceTag。

PowerShell V4 的工具

範例 1：從 Amazon S3 儲存貯體和 Amazon DynamoDB 資料表移除標籤索引鍵「階段」和「版本」，以及相關聯的值。對服務進行單一呼叫以移除標籤。在移除標籤之前，Cmdlet 會提示您進行確認。若要略過確認，請新增 -Force 參數。

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

Remove-RGTResourceTag -ResourceARNList $arn1,$arn2 -TagKey "stage","version"
```

範例 2：從 Amazon S3 儲存貯體和 Amazon DynamoDB 資料表中移除標籤索引鍵「階段」和「版本」，以及相關聯的值。對服務進行兩個呼叫，每個資源 ARN 一個呼叫會輸送到 cmdlet。每次呼叫之前，Cmdlet 都會提示您進行確認。若要略過確認，請新增 -Force 參數。

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"
```

```
$arn1,$arn2 | Remove-RGTResourceTag -TagKey "stage","version"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [UntagResources](#)。

使用 Tools for PowerShell V4 的 Route 53 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 Route 53 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執程式碼的指示。

主題

- [動作](#)

動作

Edit-R53ResourceRecordSet

以下程式碼範例顯示如何使用 Edit-R53ResourceRecordSet。

PowerShell V4 的工具

範例 1：此範例會建立 `www.example.com` 的 A 記錄，並將 `test.example.com` 的 A 記錄從 192.0.2.3 變更為 192.0.2.1。請注意，變更 TXT 類型記錄的值必須以雙引號表示。如需詳細資訊，請參閱 Amazon Route 53 文件。您可以使用 `Get-R53Change` cmdlet 輪詢，以判斷變更何時完成。

```
$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

```

範例 2：此範例示範如何建立別名資源紀錄集。'Z222222222' 是您要建立別名資源紀錄集的 Amazon Route 53 託管區域的 ID。'example.com' 是您要建立別名的區域頂點，而 'www.example.com' 是您也想要建立別名的子網域。'Z111111111111' 是負載平衡器託管區域 ID 的範例，而 'example-load-balancer-1111111111.us-east-1.elb.amazonaws.com' 是 Amazon Route 53 回應 example.com 和 www.example.com 查詢的負載平衡器網域名稱範例。如需詳細資訊，請參閱 Amazon Route 53 文件。您可以使用 Get-R53Change cmdlet 輪詢，以判斷變更何時完成。

```

$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 = "Z111111111111"
$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 = "Z11111111111111"
$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

```

範例 3：此範例會為 `www.example.com` 建立兩個 A 記錄。四分之一的時間 ($1/(1+3)$)，Amazon Route 53 會使用第一個資源記錄集 (192.0.2.9 和 192.0.2.10) 的兩個值來回應 `www.example.com` 的查詢。四分之三的時間 ($3/(1+3)$) Amazon Route 53 會使用第二個資源記錄集 (192.0.2.11 和 192.0.2.12) 的兩個值來回應 `www.example.com` 的查詢。如需詳細資訊，請參閱 Amazon Route 53 文件。您可以使用 `Get-R53Change` cmdlet 輪詢，以判斷變更何時完成。

```

$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

```

範例 4：此範例顯示如何建立加權別名資源紀錄集，假設 example.com 是您要為其建立加權別名資源紀錄集的網域。SetIdentifier 會區分兩個加權別名資源紀錄集。此元素是必要的，因為名稱和類型元素對於兩個資源紀錄集都有相同的值。Z11111111111111 和 Z33333333333333 是 DNSName 值指定之 ELB 負載平衡器的託管區域 IDs 範例。example-load-balancer-2222222222.us-east-1.elb.amazonaws.com : // 和 example-load-balancer-4444444444.us-east-1.elb.amazonaws.com 是 Elastic Load Balancing 網域的範例，Amazon Route 53 從中回應 example.com 的查詢。如需詳細資訊，請參閱 Amazon Route 53 文件。您可以使用 Get-R53Change cmdlet 輪詢，以判斷變更何時完成。

```

$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

$params = @{
    HostedZoneId="Z5555555555"
    ChangeBatch_Comment="This change batch creates two weighted alias resource
    record sets. Amazon Route 53 responds to queries for example.com with the first ELB
    domain 3/4ths of the times and the second one 1/4th of the time."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params

```

範例 5：此範例會建立兩個延遲別名資源紀錄集，一個用於美國西部（奧勒岡）區域的 ELB 負載平衡器 (us-west-2)，另一個用於亞太區域（新加坡）區域的負載平衡器 (ap-southeast-1)。如需詳細資訊，請參閱 Amazon Route 53 文件。您可以使用 Get-R53Change cmdlet 輪詢，以判斷變更何時完成。

```

$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 = "Z11111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-2222222222.us-west-2.elb.amazonaws.com"
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "example.com"
$change2.ResourceRecordSet.Type = "A"

```

```
$change2.ResourceRecordSet.SetIdentifier = "Singapore load balancer 1"
$change2.ResourceRecordSet.Region = ap-southeast-1
$change2.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change2.ResourceRecordSet.AliasTarget.HostedZoneId = "Z222222222222"
$change2.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-111111111.ap-southeast-1.elb.amazonaws.com"
$change2.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$params = @{
    HostedZoneId="Z5555555555"
    ChangeBatch_Comment="This change batch creates two latency resource record
sets, one for the US West (Oregon) region and one for the Asia Pacific (Singapore)
region."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ChangeResourceRecordSets](#)。

Get-R53AccountLimit

以下程式碼範例顯示如何使用 Get-R53AccountLimit。

PowerShell V4 的工具

範例 1：此範例會傳回可以使用目前帳戶建立的託管區域數目上限。

```
Get-R53AccountLimit -Type MAX_HOSTED_ZONES_BY_OWNER
```

輸出：

```
15
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetAccountLimit](#)。

Get-R53CheckerIpRanges

以下程式碼範例顯示如何使用 Get-R53CheckerIpRanges。

PowerShell V4 的工具

範例 1：此範例會傳回 Route53 運作狀態檢查程式CIDRs

```
Get-R53CheckerIpRanges
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetCheckerIpRanges](#)。

Get-R53HostedZone

以下程式碼範例顯示如何使用 Get-R53HostedZone。

PowerShell V4 的工具

範例 1：傳回 ID Z1D633PJN98FT9 託管區域的詳細資訊。

```
Get-R53HostedZone -Id Z1D633PJN98FT9
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetHostedZone](#)。

Get-R53HostedZoneCount

以下程式碼範例顯示如何使用 Get-R53HostedZoneCount。

PowerShell V4 的工具

範例 1：傳回目前的公有和私有託管區域總數 AWS 帳戶。

```
Get-R53HostedZoneCount
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetHostedZoneCount](#)。

Get-R53HostedZoneLimit

以下程式碼範例顯示如何使用 Get-R53HostedZoneLimit。

PowerShell V4 的工具

範例 1：此範例會傳回可在指定託管區域中建立之記錄數目上限的限制。

```
Get-R53HostedZoneLimit -HostedZoneId Z3MEQ8T7HAAAAF -Type MAX_RRSETS_BY_ZONE
```

輸出：

```
5
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetHostedZoneLimit](#)。

Get-R53HostedZoneList

以下程式碼範例顯示如何使用 Get-R53HostedZoneList。

PowerShell V4 的工具

範例 1：輸出所有公有和私有託管區域。

```
Get-R53HostedZoneList
```

範例 2：輸出與具有 ID NZ8X2CISAMPLE 的可重複使用委派集相關聯的所有託管區域

```
Get-R53HostedZoneList -DelegationSetId NZ8X2CISAMPLE
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [ListHostedZones](#)。

Get-R53HostedZonesByName

以下程式碼範例顯示如何使用 Get-R53HostedZonesByName。

PowerShell V4 的工具

範例 1：依網域名稱，以 ASCII 順序傳回所有公有和私有託管區域。

```
Get-R53HostedZonesByName
```

範例 2：依網域名稱的 ASCII 順序傳回公有和私有託管區域，從指定的 DNS 名稱開始。

```
Get-R53HostedZonesByName -DnsName example2.com
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListHostedZonesByName](#)。

Get-R53QueryLoggingConfigList

以下程式碼範例顯示如何使用 Get-R53QueryLoggingConfigList。

PowerShell V4 的工具

範例 1：此範例會傳回與目前 相關聯的 DNS 查詢記錄的所有組態 AWS 帳戶。

```
Get-R53QueryLoggingConfigList
```

輸出：

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:*
e38ddda-ceb6-45c1-8cb7-f0ae56aaaa2b	Z3MEQ8T7AAA1BF	arn:aws:logs:us-east-1:111111111112:log-group:/aws/route53/example3.com:*

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListQueryLoggingConfigs](#)。

Get-R53ReusableDelegationSet

以下程式碼範例顯示如何使用 Get-R53ReusableDelegationSet。

PowerShell V4 的工具

範例 1：此範例會擷取指定委派集的相關資訊，包括指派給委派集的四個名稱伺服器。

```
Get-R53ReusableDelegationSet -Id N23DS9X4AYEAAA
```

輸出：

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}

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetReusableDelegationSet](#)。

New-R53HostedZone

以下程式碼範例顯示如何使用 New-R53HostedZone。

PowerShell V4 的工具

範例 1：建立名為 'example.com' 的新託管區域，與可重複使用的委派集相關聯。請注意，您必須提供 CallerReference 參數的值，以便需要在必要時重試的請求，而不會有執行操作兩次的風險。由於託管區域是在 VPC 中建立，因此會自動私有，而且您不應該設定 -HostedZoneConfig_PrivateZone 參數。

```
$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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateHostedZone](#)。

New-R53QueryLoggingConfig

以下程式碼範例顯示如何使用 New-R53QueryLoggingConfig。

PowerShell V4 的工具

範例 1：此範例會為指定的託管區域建立新的 Route53 DNS 查詢記錄組態。Amazon Route53 會將 DNS 查詢日誌發佈至指定的 Cloudwatch 日誌群組。

```
New-R53QueryLoggingConfig -HostedZoneId Z3MEQ8T7HAAAAF -CloudWatchLogsLogGroupArn
arn:aws:logs:us-east-1:111111111111:log-group:/aws/route53/example.com:*
```

輸出：

QueryLoggingConfig	Location
-----	-----

```
Amazon.Route53.Model.QueryLoggingConfig https://route53.amazonaws.com/2013-04-01/
queryloggingconfig/ee5aaa95-4e03-4fdc-9d28-9e24ddaaaaa3
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [CreateQueryLoggingConfig](#)。

New-R53ReusableDelegationSet

以下程式碼範例顯示如何使用 New-R53ReusableDelegationSet。

PowerShell V4 的工具

範例 1：此範例會建立一組可重複使用的委派，其中包含 4 個名稱伺服器，可供多個託管區域重複使用。

```
New-R53ReusableDelegationSet -CallerReference testcallerreference
```

輸出：

DelegationSet	Location
-----	-----
Amazon.Route53.Model.DelegationSet	https://route53.amazonaws.com/2013-04-01/
delegationset/N23DS9XAAAAAXM	

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateReusableDelegationSet](#)。

Register-R53VPCWithHostedZone

以下程式碼範例顯示如何使用 Register-R53VPCWithHostedZone。

PowerShell V4 的工具

範例 1：此範例會將指定的 VPC 與私有託管區域建立關聯。

```
Register-R53VPCWithHostedZone -HostedZoneId Z3MEQ8T7HAAAAF -VPC_VPCId vpc-f1b9aaaa -
VPC_VPCRegion us-east-1
```

輸出：

Id	Status	SubmittedAt	Comment

```
--  
-----  
/change/C3SCAAA633Z6DX PENDING 01/28/2020 19:32:02
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet 參考》(V4) 中的 [AssociateVPCWithHostedZone](#)。

Remove-R53HostedZone

以下程式碼範例顯示如何使用 Remove-R53HostedZone。

PowerShell V4 的工具

範例 1：刪除具有指定 ID 的託管區域。除非您新增 -Force 切換參數，否則在命令繼續之前會提示您進行確認。

```
Remove-R53HostedZone -Id Z1PA6795UKMFR9
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteHostedZone](#)。

Remove-R53QueryLoggingConfig

以下程式碼範例顯示如何使用 Remove-R53QueryLoggingConfig。

PowerShell V4 的工具

範例 1：此範例會移除 DNS 查詢記錄的指定組態。

```
Remove-R53QueryLoggingConfig -Id ee528e95-4e03-4fdc-9d28-9e24daaa20063
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteQueryLoggingConfig](#)。

Remove-R53ReusableDelegationSet

以下程式碼範例顯示如何使用 Remove-R53ReusableDelegationSet。

PowerShell V4 的工具

範例 1：此範例會刪除指定的可重複使用委派集。

```
Remove-R53ReusableDelegationSet -Id N23DS9X4AYAAAM
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteReusableDelegationSet](#)。

Unregister-R53VPCFromHostedZone

以下程式碼範例顯示如何使用 Unregister-R53VPCFromHostedZone。

PowerShell V4 的工具

範例 1：此範例會取消指定的 VPC 與私有託管區域的關聯。

```
Unregister-R53VPCFromHostedZone -HostedZoneId Z3MEQ8T7HAAAF -VPC_VPCId vpc-f1b9aaaa  
-VPC_VPCRegion us-east-1
```

輸出：

Id	Status	SubmittedAt	Comment
--	-----	-----	-----
/change/C2XFCAAAA9HKZG	PENDING	01/28/2020 10:35:55	

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DisassociateVPCFromHostedZone](#)。

Update-R53HostedZoneComment

以下程式碼範例顯示如何使用 Update-R53HostedZoneComment。

PowerShell V4 的工具

範例 1：此命令會更新指定託管區域的註解。

```
Update-R53HostedZoneComment -Id Z385PDS6AAAAAR -Comment "This is my first hosted  
zone"
```

輸出：

```
Id : /hostedzone/Z385PDS6AAAAAR
```

```
Name                : example.com.  
CallerReference      : C5B55555-7147-EF04-8341-69131E805C89  
Config              : Amazon.Route53.Model.HostedZoneConfig  
ResourceRecordSetCount : 9  
LinkedService       :
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UpdateHostedZoneComment](#)。

使用 Tools for PowerShell V4 的 Amazon S3 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 Amazon S3 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Copy-S3object

以下程式碼範例顯示如何使用 Copy-S3object。

PowerShell V4 的工具

範例 1：此命令會將物件 "sample.txt" 從儲存貯體 "test-files" 複製到相同的儲存貯體，但新索引鍵為 "sample-copy.txt"。

```
Copy-S3object -BucketName amzn-s3-demo-bucket -Key sample.txt -DestinationKey  
sample-copy.txt
```

範例 2：此命令會將物件 "sample.txt" 從儲存貯體 "test-files" 複製到儲存貯體 "backup-files"，金鑰為 "sample-copy.txt"。

```
Copy-S3Object -BucketName amzn-s3-demo-source-bucket -Key sample.txt -DestinationKey
sample-copy.txt -DestinationBucket amzn-s3-demo-destination-bucket
```

範例 3：此命令會將物件 "sample.txt" 從儲存貯體 "test-files" 下載至名為 "local-sample.txt" 的本機檔案。

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-
sample.txt
```

範例 4：將單一物件下載至指定的檔案。下載的檔案位於 c:\downloads\data\archive.zip

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key data/archive.zip -LocalFolder c:
\downloads
```

範例 5：將符合指定金鑰字首的所有物件下載至本機資料夾。相對金鑰階層會保留為整體下載位置中的子資料夾。

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data -LocalFolder c:
\downloads
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CopyObject](#)。

Get-S3ACL

以下程式碼範例顯示如何使用 Get-S3ACL。

PowerShell V4 的工具

範例 1：命令會取得 S3 物件之物件擁有者的詳細資訊。

```
Get-S3ACL -BucketName 'amzn-s3-demo-bucket' -key 'initialize.ps1' -Select
AccessControlList.Owner
```

輸出：

```
DisplayName Id
----- --
testusername 9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [GetACL](#)。

Get-S3Bucket

以下程式碼範例顯示如何使用 Get-S3Bucket。

PowerShell V4 的工具

範例 1：此命令會傳回所有 S3 儲存貯體。

```
Get-S3Bucket
```

範例 2：此命令會傳回名為 "test-files" 的儲存貯體

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListBuckets](#)。

Get-S3BucketAccelerateConfiguration

以下程式碼範例顯示如何使用 Get-S3BucketAccelerateConfiguration。

PowerShell V4 的工具

範例 1：如果為指定的儲存貯體啟用傳輸加速設定，此命令會傳回已啟用的值。

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

輸出：

```
Value
-----
Enabled
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GetBucketAccelerateConfiguration](#)。

Get-S3BucketAnalyticsConfiguration

以下程式碼範例顯示如何使用 Get-S3BucketAnalyticsConfiguration。

PowerShell V4 的工具

範例 1：此命令會傳回指定 S3 儲存貯體中名為 'testfilter' 的分析篩選條件詳細資訊。

```
Get-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId  
'testfilter'
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GetBucketAnalyticsConfiguration](#)。

Get-S3BucketAnalyticsConfigurationList

以下程式碼範例顯示如何使用 Get-S3BucketAnalyticsConfigurationList。

PowerShell V4 的工具

範例 1：此命令會傳回指定 S3 儲存貯體的前 100 個分析組態。

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListBucketAnalyticsConfigurations](#)。

Get-S3BucketEncryption

以下程式碼範例顯示如何使用 Get-S3BucketEncryption。

PowerShell V4 的工具

範例 1：此命令會傳回與指定儲存貯體相關聯的所有伺服器端加密規則。

```
Get-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GetBucketEncryption](#)。

Get-S3BucketInventoryConfiguration

以下程式碼範例顯示如何使用 Get-S3BucketInventoryConfiguration。

PowerShell V4 的工具

範例 1：此命令會傳回指定 S3 儲存貯體之名為「testinventory」的庫存詳細資訊。

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId  
'testinventory'
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet 參考》(V4) 中的 [GetBucketInventoryConfiguration](#)。

Get-S3BucketInventoryConfigurationList

以下程式碼範例顯示如何使用 Get-S3BucketInventoryConfigurationList。

PowerShell V4 的工具

範例 1：此命令會傳回指定 S3 儲存貯體的前 100 個庫存組態。

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListBucketInventoryConfigurations](#)。

Get-S3BucketLocation

以下程式碼範例顯示如何使用 Get-S3BucketLocation。

PowerShell V4 的工具

範例 1：如果存在限制，此命令會傳回儲存貯體 'amzn-s3-demo-bucket' 的位置限制。

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

輸出：

```
Value  
-----  
ap-south-1
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetBucketLocation](#)。

Get-S3BucketLogging

以下程式碼範例顯示如何使用 Get-S3BucketLogging。

PowerShell V4 的工具

範例 1：此命令會傳回指定儲存貯體的記錄狀態。

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

輸出：

TargetBucketName	Grants	TargetPrefix
testbucket1	{}	testprefix

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetBucketLogging](#)。

Get-S3BucketMetricsConfiguration

以下程式碼範例顯示如何使用 Get-S3BucketMetricsConfiguration。

PowerShell V4 的工具

範例 1：此命令會傳回指定 S3 儲存貯體名為 'testfilter' 的指標篩選條件詳細資訊。

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId 'testfilter'
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GetBucketMetricsConfiguration](#)。

Get-S3BucketNotification

以下程式碼範例顯示如何使用 Get-S3BucketNotification。

PowerShell V4 的工具

範例 1：此範例會擷取指定儲存貯體的通知組態

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty
TopicConfigurations
```

輸出：

```
Id      Topic
--      -
mimo    arn:aws:sns:eu-west-1:123456789012:topic-1
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GetBucketNotification](#)。

Get-S3BucketPolicy

以下程式碼範例顯示如何使用 Get-S3BucketPolicy。

PowerShell V4 的工具

範例 1：此命令會輸出與指定 S3 儲存貯體相關聯的儲存貯體政策。

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GetBucketPolicy](#)。

Get-S3BucketPolicyStatus

以下程式碼範例顯示如何使用 Get-S3BucketPolicyStatus。

PowerShell V4 的工具

範例 1：此命令會傳回指定 S3 儲存貯體的政策狀態，指出儲存貯體是否為公有。

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GetBucketPolicyStatus](#)。

Get-S3BucketReplication

以下程式碼範例顯示如何使用 Get-S3BucketReplication。

PowerShell V4 的工具

範例 1：傳回在名為 'amzn-s3-demo-bucket' 的儲存貯體上設定的複寫組態資訊。

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GetBucketReplication](#)。

Get-S3BucketRequestPayment

以下程式碼範例顯示如何使用 Get-S3BucketRequestPayment。

PowerShell V4 的工具

範例 1：傳回名為 'amzn-s3-demo-bucket' 之儲存貯體的請求付款組態。根據預設，儲存貯體擁有者會支付從儲存貯體下載的費用。

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetBucketRequestPayment](#)。

Get-S3BucketTagging

以下程式碼範例顯示如何使用 Get-S3BucketTagging。

PowerShell V4 的工具

範例 1：此命令會傳回與指定儲存貯體相關聯的所有標籤。

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetBucketTagging](#)。

Get-S3BucketVersioning

以下程式碼範例顯示如何使用 Get-S3BucketVersioning。

PowerShell V4 的工具

範例 1：此命令會傳回與指定儲存貯體相關的版本控制狀態。

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetBucketVersioning](#)。

Get-S3BucketWebsite

以下程式碼範例顯示如何使用 Get-S3BucketWebsite。

PowerShell V4 的工具

範例 1：此命令會傳回指定 S3 儲存貯體靜態網站組態的詳細資訊。

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetBucketWebsite](#)。

Get-S3CORSConfiguration

以下程式碼範例顯示如何使用 Get-S3CORSConfiguration。

PowerShell V4 的工具

範例 1：此命令會傳回物件，其中包含對應至指定 S3 儲存貯體的所有 CORS 組態規則。

```
Get-S3CORSConfiguration -BucketName 'amzn-s3-demo-bucket' -Select  
Configuration.Rules
```

輸出：

```
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 : {}
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetCORSConfiguration](#)。

Get-S3LifecycleConfiguration

以下程式碼範例顯示如何使用 Get-S3LifecycleConfiguration。

PowerShell V4 的工具

範例 1：此範例會擷取儲存貯體的生命週期組態。

```
Get-S3LifecycleConfiguration -BucketName amzn-s3-demo-bucket
```

輸出：

```
Rules
-----
{Remove-in-150-days, Archive-to-Glacier-in-30-days}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetLifecycleConfiguration](#)。

Get-S3Object

以下程式碼範例顯示如何使用 Get-S3Object。

PowerShell V4 的工具

範例 1：此命令會擷取儲存貯體 "test-files" 中所有項目的相關資訊。

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

範例 2：此命令會從儲存貯體 "test-files" 擷取項目 "sample.txt" 的相關資訊。

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

範例 3：此命令會從儲存貯體 "test-files" 擷取字首為 "sample" 之所有項目的相關資訊。

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListObjects](#)。

Get-S3ObjectLockConfiguration

以下程式碼範例顯示如何使用 Get-S3ObjectLockConfiguration。

PowerShell V4 的工具

範例 1：如果為指定的 S3 儲存貯體啟用物件鎖定組態，此命令會傳回值 'Enabled'。

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select  
ObjectLockConfiguration.ObjectLockEnabled
```

輸出：

```
Value  
-----
```

Enabled

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetObjectLockConfiguration](#)。

Get-S3ObjectMetadata

以下程式碼範例顯示如何使用 Get-S3ObjectMetadata。

PowerShell V4 的工具

範例 1：此命令會傳回指定 S3 儲存貯體中金鑰為 'ListTrusts.txt' 的物件中繼資料。

```
Get-S3ObjectMetadata -BucketName 'amzn-s3-demo-bucket' -Key 'ListTrusts.txt'
```

輸出：

```
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         :
```


- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetObjectMetadata](#)。

Get-S3ObjectRetention

以下程式碼範例顯示如何使用 Get-S3ObjectRetention。

PowerShell V4 的工具

範例 1：命令會傳回模式和日期，直到保留物件為止。

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetObjectRetention](#)。

Get-S3ObjectTagSet

以下程式碼範例顯示如何使用 Get-S3ObjectTagSet。

PowerShell V4 的工具

範例 1：範例會傳回與特定 S3 儲存貯體上存在之物件相關聯的標籤。

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

輸出：

```
Key  Value
---  -
test value
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetObjectTagging](#)。

Get-S3PreSignedURL

以下程式碼範例顯示如何使用 Get-S3PreSignedURL。

PowerShell V4 的工具

範例 1：命令會傳回指定金鑰和過期日期的預先簽章 URL。

```
Get-S3PreSignedURL -BucketName 'amzn-s3-demo-bucket' -Key 'testkey' -Expires  
'2023-11-16'
```

範例 2：命令會傳回具有指定金鑰和過期日期的目錄儲存貯體的預先簽章 URL。

```
[Amazon.AWSConfigsS3]::UseSignatureVersion4 = $true  
Get-S3PreSignedURL -BucketName amzn-s3-demo-bucket--usw2-az1--x-s3 -Key  
'testkey' -Expire '2023-11-17'
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetPreSignedURL](#)。

Get-S3PublicAccessBlock

以下程式碼範例顯示如何使用 Get-S3PublicAccessBlock。

PowerShell V4 的工具

範例 1：命令會傳回指定 S3 儲存貯體的公有存取區塊組態。

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetPublicAccessBlock](#)。

Get-S3Version

以下程式碼範例顯示如何使用 Get-S3Version。

PowerShell V4 的工具

範例 1：此命令會傳回指定 S3 儲存貯體中所有物件版本的中繼資料。

```
Get-S3Version -BucketName 'amzn-s3-demo-bucket'
```

輸出：

```

IsTruncated      : False
KeyMarker        :
VersionIdMarker  :
NextKeyMarker    :
NextVersionIdMarker :
Versions         : {EC2.txt, EC2MicrosoftWindowsGuide.txt, ListDirectories.json,
  ListTrusts.json}
Name             : amzn-s3-demo-bucket
Prefix           :
MaxKeys          : 1000
CommonPrefixes   : {}
Delimiter        :

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListVersions](#)。

New-S3Bucket

以下程式碼範例顯示如何使用 New-S3Bucket。

PowerShell V4 的工具

範例 1：此命令會建立新的私有儲存貯體，名為 "sample-bucket"。

```
New-S3Bucket -BucketName amzn-s3-demo-bucket
```

範例 2：此命令會建立名為 "sample-bucket" 且具有讀寫許可的新儲存貯體。

```
New-S3Bucket -BucketName amzn-s3-demo-bucket -PublicReadWrite
```

範例 3：此命令會建立名為「sample-bucket」且具有唯讀許可的新儲存貯體。

```
New-S3Bucket -BucketName amzn-s3-demo-bucket -PublicReadOnly
```

範例 4：此命令會使用 PutBucketConfiguration 建立新的目錄儲存貯體，名為 "amzn-s3-demo-bucket--use1-az5--x-s3"。

```

$bucketConfiguration = @{
    BucketInfo = @{
        DataRedundancy = 'SingleAvailabilityZone'
    }
}

```

```

        Type = 'Directory'
    }
    Location = @{
        Name = 'usw2-az1'
        Type = 'AvailabilityZone'
    }
}
New-S3Bucket -BucketName amzn-s3-demo-bucket--usw2-az1--x-s3 -BucketConfiguration
$bucketConfiguration -Region us-west-2

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [PutBucket](#)。

Read-S3Object

以下程式碼範例顯示如何使用 Read-S3Object。

PowerShell V4 的工具

範例 1：此命令會從儲存貯體 "test-files" 擷取項目 "sample.txt"，並將其儲存至目前位置中名為 "local-sample.txt" 的檔案。在呼叫此命令之前，檔案「local-sample.txt」不必存在。

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt
```

範例 2：此命令會從儲存貯體 "test-files" 擷取虛擬目錄 "DIR"，並將其儲存至目前位置中名為 "Local-DIR" 的資料夾。呼叫此命令之前，資料夾「Local-DIR」不必存在。

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

範例 3：將所有金鑰以 '.json' 結尾的物件，從儲存貯體名稱具有 'config' 的儲存貯體下載到指定資料夾中的檔案。物件金鑰用於設定檔案名稱。

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -
like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [GetObject](#)。

Remove-S3Bucket

以下程式碼範例顯示如何使用 Remove-S3Bucket。

PowerShell V4 的工具

範例 1：此命令會從儲存貯體「test-files」移除所有物件和物件版本，然後刪除儲存貯體。命令會先提示確認再繼續。新增 -Force 切換以隱藏確認。請注意，無法刪除不是空的儲存貯體。

```
Remove-S3Bucket -BucketName amzn-s3-demo-bucket -DeleteBucketContent
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteBucket](#)。

Remove-S3BucketAnalyticsConfiguration

以下程式碼範例顯示如何使用 Remove-S3BucketAnalyticsConfiguration。

PowerShell V4 的工具

範例 1：命令會移除指定 S3 儲存貯體中名為 'testfilter' 的分析篩選條件。

```
Remove-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId  
'testfilter'
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteBucketAnalyticsConfiguration](#)。

Remove-S3BucketEncryption

以下程式碼範例顯示如何使用 Remove-S3BucketEncryption。

PowerShell V4 的工具

範例 1：這會停用針對提供的 S3 儲存貯體啟用的加密。

```
Remove-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteBucketEncryption](#)。

Remove-S3BucketInventoryConfiguration

以下程式碼範例顯示如何使用 Remove-S3BucketInventoryConfiguration。

PowerShell V4 的工具

範例 1：此命令會移除與指定 S3 儲存貯體對應的名為 'testInventoryName' 的庫存。

```
Remove-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId  
'testInventoryName'
```

輸出：

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-S3BucketInventoryConfiguration  
(DeleteBucketInventoryConfiguration)" on target "amzn-s3-demo-bucket".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): Y
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteBucketInventoryConfiguration](#)。

Remove-S3BucketMetricsConfiguration

以下程式碼範例顯示如何使用 Remove-S3BucketMetricsConfiguration。

PowerShell V4 的工具

範例 1：命令會移除指定 S3 儲存貯體中名為 'testmetrics' 的指標篩選條件。

```
Remove-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId  
'testmetrics'
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteBucketMetricsConfiguration](#)。

Remove-S3BucketPolicy

以下程式碼範例顯示如何使用 Remove-S3BucketPolicy。

PowerShell V4 的工具

範例 1：命令會移除與指定 S3 儲存貯體相關聯的儲存貯體政策。

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteBucketPolicy](#)。

Remove-S3BucketReplication

以下程式碼範例顯示如何使用 Remove-S3BucketReplication。

PowerShell V4 的工具

範例 1：刪除與名為 'amzn-s3-demo-bucket' 的儲存貯體相關聯的複寫組態。請注意，此操作需要 s3:DeleteReplicationConfiguration 動作的許可。在操作繼續之前，系統會提示您進行確認 - 若要隱藏確認，請使用 -Force 切換。

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteBucketReplication](#)。

Remove-S3BucketTagging

以下程式碼範例顯示如何使用 Remove-S3BucketTagging。

PowerShell V4 的工具

範例 1：此命令會移除與指定 S3 儲存貯體相關聯的所有標籤。

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

輸出：

```
Confirm
```

```
Are you sure you want to perform this action?  
Performing the operation "Remove-S3BucketTagging (DeleteBucketTagging)" on target  
"amzn-s3-demo-bucket".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): Y
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteBucketTagging](#)。

Remove-S3BucketWebsite

以下程式碼範例顯示如何使用 Remove-S3BucketWebsite。

PowerShell V4 的工具

範例 1：此命令會停用指定 S3 儲存貯體的靜態網站託管屬性。

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

輸出：

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-S3BucketWebsite (DeleteBucketWebsite)" on target  
"amzn-s3-demo-bucket".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): Y
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteBucketWebsite](#)。

Remove-S3CORSConfiguration

以下程式碼範例顯示如何使用 Remove-S3CORSConfiguration。

PowerShell V4 的工具

範例 1：此命令會移除指定 S3 儲存貯體的 CORS 組態。

```
Remove-S3CORSConfiguration -BucketName 'amzn-s3-demo-bucket'
```


輸出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3CORSConfiguration (DeleteCORSConfiguration)" on
target "amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"): Y
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteCORSConfiguration](#)。

Remove-S3LifecycleConfiguration

以下程式碼範例顯示如何使用 Remove-S3LifecycleConfiguration。

PowerShell V4 的工具

範例 1：命令會移除指定 S3 儲存貯體的所有生命週期規則。

```
Remove-S3LifecycleConfiguration -BucketName 'amzn-s3-demo-bucket'
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteLifecycleConfiguration](#)。

Remove-S3MultipartUpload

以下程式碼範例顯示如何使用 Remove-S3MultipartUpload。

PowerShell V4 的工具

範例 1：此命令中止 5 天前建立的分段上傳。

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -DaysBefore 5
```

範例 2：此命令中止 2014 年 1 月 2 日之前建立的分段上傳。

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate "Thursday,
January 02, 2014"
```

範例 3：此命令中止 2014 年 1 月 2 日之前建立的分段上傳，時間是 10:45:37。

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate "2014/01/02 10:45:37"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [AbortMultipartUpload](#)。

Remove-S3Object

以下程式碼範例顯示如何使用 Remove-S3Object。

PowerShell V4 的工具

範例 1：此命令會從儲存貯體 "test-files" 中移除物件 "sample.txt"。在命令執行之前，系統會提示您進行確認；若要隱藏提示，請使用 -Force 切換。

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

範例 2：此命令會將指定版本的物件 "sample.txt" 從儲存貯體 "test-files" 中移除，假設儲存貯體已設定為啟用物件版本。

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId HLBxnx6V9omT6AQYVpks8mmFKQcejpqt
```

範例 3：此命令會將儲存貯體 "test-files" 中的物件 "sample1.txt"、"sample2.txt" 和 "sample3.txt" 移除為單一批次操作。無論刪除成功或錯誤狀態為何，服務回應都會列出處理的所有金鑰。若要僅取得無法由服務處理之金鑰的錯誤，請新增 -ReportErrorsOnly 參數（也可以使用別名 -Quiet 指定此參數）。

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt", "sample2.txt", "sample3.txt" )
```

範例 4：此範例使用內嵌表達式搭配 -KeyCollection 參數，以取得要刪除之物件的金鑰。Get-S3Object 會傳回 Amazon.S3.Model.S3Object 執行個體的集合，每個執行個體都有識別物件的類型字串索引鍵成員。

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object "test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

範例 5：此範例會取得儲存貯體中具有金鑰字首「字首/子字首」的所有物件，並將其刪除。請注意，傳入的物件會一次處理一個。對於大型集合，請考慮將集合傳遞至 cmdlet 的 -InputObject (別名 -S3ObjectCollection) 參數，讓刪除以批次形式發生，並對服務進行單一呼叫。

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |  
Remove-S3Object -Force
```

範例 6：此範例會將代表刪除標記的 Amazon.S3.Model.S3ObjectVersion 執行個體集合輸送至 cmdlet 以進行刪除。請注意，傳入的物件會一次處理一個。對於大型集合，請考慮將集合傳遞至 cmdlet 的 -InputObject (別名 -S3ObjectCollection) 參數，讓刪除以批次形式發生，並對服務進行單一呼叫。

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where  
{$_ .IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

範例 7：此指令碼說明如何透過建構要與 -KeyAndVersionCollection 參數搭配使用的物件陣列，執行一組物件的批次刪除 (在此案例中為刪除標記)。

```
$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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteObjects](#)。

Remove-S3ObjectTagSet

以下程式碼範例顯示如何使用 Remove-S3ObjectTagSet。

PowerShell V4 的工具

範例 1：此命令會移除與指定 S3 儲存貯體中金鑰為 'testfile.txt' 之物件相關聯的所有標籤。

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -Select  
'^Key'
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteObjectTagging](#)。

Remove-S3PublicAccessBlock

以下程式碼範例顯示如何使用 Remove-S3PublicAccessBlock。

PowerShell V4 的工具

範例 1：此命令會關閉指定儲存貯體的封鎖公開存取設定。

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select
'^BucketName'
```

輸出：

```
amzn-s3-demo-bucket
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeletePublicAccessBlock](#)。

Set-S3BucketEncryption

以下程式碼範例顯示如何使用 Set-S3BucketEncryption。

PowerShell V4 的工具

範例 1：此命令會在指定的儲存貯體上使用 Amazon S3 受管金鑰 (SSE-S3) 啟用預設 AES256 伺服器端加密。

```
$Encryptionconfig = @{{ServerSideEncryptionByDefault =  
    @{{ServerSideEncryptionAlgorithm = "AES256"}}}  
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -  
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [PutBucketEncryption](#)。

Test-S3Bucket

以下程式碼範例顯示如何使用 Test-S3Bucket。

PowerShell V4 的工具

範例 1：如果儲存貯體存在，則此命令會傳回 True，否則傳回 False。即使儲存貯體不屬於使用者，命令也會傳回 True。

```
Test-S3Bucket -BucketName amzn-s3-demo-bucket
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [Test-S3Bucket](#)。

Write-S3BucketAccelerateConfiguration

以下程式碼範例顯示如何使用 Write-S3BucketAccelerateConfiguration。

PowerShell V4 的工具

範例 1：此命令會啟用指定 S3 儲存貯體的傳輸加速。

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')  
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AccelerateConfiguration_Status $statusVal
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [PutBucketAccelerateConfiguration](#)。

Write-S3BucketNotification

以下程式碼範例顯示如何使用 Write-S3BucketNotification。

PowerShell V4 的工具

範例 1：此範例會設定 S3 事件 ObjectRemovedDelete 的 SNS 主題組態，並啟用指定 s3 儲存貯體的通知

```
$topic = [Amazon.S3.Model.TopicConfiguration] @{
    Id = "delete-event"
    Topic = "arn:aws:sns:eu-west-1:123456789012:topic-1"
    Event = [Amazon.S3.EventType]::ObjectRemovedDelete
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -TopicConfiguration
$topic
```

範例 2：此範例會為傳送至 Lambda 函數的指定儲存貯體啟用 ObjectCreatedAll 的通知。

```
$lambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = "s3:ObjectCreated:*"
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:rdplock"
    Id = "ObjectCreated-Lambda"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".pem"}
            )
        }
    }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $lambdaConfig
```

範例 3：此範例會根據不同的金鑰尾碼建立 2 個不同的 Lambda 組態，並在單一命令中同時設定兩者。

```
#Lambda Config 1

$firstLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = "s3:ObjectCreated:*"
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifynet"
    Id = "ObjectCreated-dada-ps1"
```

```

Filter = @{
    S3KeyFilter = @{
        FilterRules = @(
            @{Name="Prefix";Value="dada"}
            @{Name="Suffix";Value=".ps1"}
        )
    }
}

#Lambda Config 2

$secondlambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = [Amazon.S3.EventType]::ObjectCreatedAll
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifyssm"
    Id = "ObjectCreated-dada-json"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".json"}
            )
        }
    }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $firstLambdaConfig,$secondlambdaConfig

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [PutBucketNotification](#)。

Write-S3BucketReplication

以下程式碼範例顯示如何使用 Write-S3BucketReplication。

PowerShell V4 的工具

範例 1：此範例會設定複寫組態，其中包含啟用複寫至 'amzn-s3-demo-bucket' 儲存貯體 'amzn-s3-demo-bucket' 中金鑰名稱字首為 "TaxDocs" 的任何新物件。

```

$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"

```

```
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

範例 2：此範例設定具有多個規則的複寫組態，可將任何以金鑰名稱字首 "TaxDocs" 或 "OtherDocs" 建立的新物件複寫至 'amzn-s3-demo-bucket' 儲存貯體。金鑰字首不得重疊。

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-bucket" }

$rule2 = New-Object Amazon.S3.Model.ReplicationRule
$rule2.ID = "Rule-2"
$rule2.Status = "Enabled"
$rule2.Prefix = "OtherDocs"
$rule2.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1,$rule2
}

Write-S3BucketReplication @params
```

範例 3：此範例會更新指定儲存貯體上的複寫組態，以停用控制金鑰名稱字首為 "TaxDocs" 之物件複寫至儲存貯體 'amzn-s3-demo-bucket' 的規則。

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
```



```
$rule1.Status = "Disabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [PutBucketReplication](#)。

Write-S3BucketRequestPayment

以下程式碼範例顯示如何使用 Write-S3BucketRequestPayment。

PowerShell V4 的工具

範例 1：更新名為 'amzn-s3-demo-bucket' 之儲存貯體的請求付款組態，以便從儲存貯體請求下載的人員需支付下載費用。根據預設，儲存貯體擁有者會支付下載費用。若要將請求付款設回預設，請對 RequestPaymentConfiguration_Payer 參數使用 'BucketOwner'。

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -
RequestPaymentConfiguration_Payer Requester
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [PutBucketRequestPayment](#)。

Write-S3BucketTagging

以下程式碼範例顯示如何使用 Write-S3BucketTagging。

PowerShell V4 的工具

範例 1：此命令會將兩個標籤套用至名為 **cloudtrail-test-2018** 的儲存貯體：索引鍵為 Stage 且值為 Test 的標籤，以及索引鍵為 Environment 且值為 Alpha 的標籤。若要驗證標籤是否已新增至儲存貯體，請執行 **Get-S3BucketTagging -BucketName bucket_name**。結果應該會

顯示您在第一個命令中套用至儲存貯體的標籤。請注意，會 **Write-S3BucketTagging** 覆寫儲存貯體上整個現有標籤集。若要新增或刪除個別標籤，請執行資源群組和標記 API cmdlet，**Add-RGTResourceTag** 以及 **Remove-RGTResourceTag**。或者，在 AWS 管理主控台中使用標籤編輯器來管理 S3 儲存貯體標籤。

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage"; Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

範例 2：此命令會將名為 **cloudtrail-test-2018** 的儲存貯體輸送至 **Write-S3BucketTagging** cmdlet。它會將標籤 Stage : Production 和 Department : Finance 套用至儲存貯體。請注意，會 **Write-S3BucketTagging** 覆寫儲存貯體上整個現有標籤集。

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging -TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department"; Value="Finance" } )
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [PutBucketTagging](#)。

Write-S3BucketVersioning

以下程式碼範例顯示如何使用 Write-S3BucketVersioning。

PowerShell V4 的工具

範例 1：命令會啟用指定 S3 儲存貯體的版本控制。

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -VersioningConfig_Status Enabled
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [PutBucketVersioning](#)。

Write-S3BucketWebsite

以下程式碼範例顯示如何使用 Write-S3BucketWebsite。

PowerShell V4 的工具

範例 1：命令會啟用指定儲存貯體的網站託管，索引文件為 'index.html'，錯誤文件為 'error.html'。

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
-WebsiteConfiguration_IndexDocumentSuffix 'index.html' -
WebsiteConfiguration_ErrorDocument 'error.html'
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [PutBucketWebsite](#)。

Write-S3LifecycleConfiguration

以下程式碼範例顯示如何使用 Write-S3LifecycleConfiguration。

PowerShell V4 的工具

範例 1：此範例會寫入/取代 \$NewRule 中提供的組態。此組態可確保限制具有指定字首和標籤值的範圍物件。

```
$NewRule = [Amazon.S3.Model.LifecycleRule] @{
    Expiration = @{
        Days= 50
    }
    Id = "Test-From-Write-cmdlet-1"
    Filter= @{
        LifecycleFilterPredicate = [Amazon.S3.Model.LifecycleAndOperator]@{
            Operands= @(
                [Amazon.S3.Model.LifecyclePrefixPredicate] @{
                    "Prefix" = "py"
                },
                [Amazon.S3.Model.LifecycleTagPredicate] @{
                    "Tag"= @{
                        "Key" = "non-use"
                        "Value" = "yes"
                    }
                }
            )
        }
    }
    "Status"= 'Enabled'
    NoncurrentVersionExpiration = @{
        NoncurrentDays = 75
    }
}
```

```
Write-S3LifecycleConfiguration -BucketName amzn-s3-demo-bucket -Configuration_Rule
$NewRule
```

範例 2：此範例使用篩選設定多個規則。\$ArchiveRule 會將要在 30 天內封存的物件設定為 Glacier，並將 120 設定為 DeepArchive。對於 'py' 字首和 tag : key 'archieved' 設定為 'yes' 的物件，\$ExpireRule 會在 150 天內過期目前和先前的版本。

```
$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" = "archieved"
                        "Value" = "yes"
                    }
                }
            )
        }
    }
    Status= 'Enabled'
    NoncurrentVersionExpiration = @{
        NoncurrentDays = 150
    }
}

$ArchiveRule = [Amazon.S3.Model.LifecycleRule] @{
    Expiration = $null
    Id = "Archive-to-Glacier-in-30-days"
    Filter= @{
        LifecycleFilterPredicate = [Amazon.S3.Model.LifecycleAndOperator]@{
            Operands= @(
                [Amazon.S3.Model.LifecyclePrefixPredicate] @{
                    "Prefix" = "py"
                },
                [Amazon.S3.Model.LifecycleTagPredicate] @{
```

```

        "Tag"= @{
            "Key" = "reviewed"
            "Value" = "yes"
        }
    }
)
}
}
Status = 'Enabled'
NoncurrentVersionExpiration = @{
    NoncurrentDays = 75
}
Transitions = @(
    @{
        Days = 30
        "StorageClass"= 'Glacier'
    },
    @{
        Days = 120
        "StorageClass"= [Amazon.S3.S3StorageClass]::DeepArchive
    }
)
}

Write-S3LifecycleConfiguration -BucketName amzn-s3-demo-bucket -Configuration_Rule
$ExpireRule,$ArchiveRule

```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [PutLifecycleConfiguration](#)。

Write-S3Object

以下程式碼範例顯示如何使用 Write-S3Object。

PowerShell V4 的工具

範例 1：此命令會將單一檔案 "local-sample.txt" 上傳至 Amazon S3，並在儲存貯體 "test-files" 中建立金鑰為 "sample.txt" 的物件。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-
sample.txt
```

範例 2：此命令會將單一檔案 "sample.txt" 上傳至 Amazon S3，在儲存貯體 "test-files" 中建立金鑰為 "sample.txt" 的物件。如果未提供 -Key 參數，則會使用檔案名稱做為 S3 物件金鑰。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

範例 3：此命令會將單一檔案 "local-sample.txt" 上傳至 Amazon S3，並在儲存貯體 "test-files" 中建立金鑰為 "prefix/to/sample.txt" 的物件。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -File .\local-sample.txt
```

範例 4：此命令會將子目錄 "Scripts" 中的所有檔案上傳至儲存貯體 "test-files"，並將通用金鑰字首 "SampleScripts" 套用至每個物件。每個上傳的檔案都有「SampleScripts/filename」的索引鍵，其中「filename」會有所不同。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix SampleScripts\
```

範例 5：此命令會將本機目錄 "Scripts" 中的所有 *.ps1 檔案上傳至儲存貯體 "test-files"，並將通用金鑰字首 "SampleScripts" 套用至每個物件。每個上傳的檔案都會有一個「SampleScripts/filename.ps1」的索引鍵，其中「filename」會有所不同。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix SampleScripts\ -SearchPattern *.ps1
```

範例 6：此命令會建立新的 S3 物件，其中包含金鑰為 'sample.txt' 的指定內容字串。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object contents"
```

範例 7：此命令會上傳指定的檔案（檔案名稱用作金鑰），並將指定的標籤套用至新物件。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

範例 8：此命令會遞迴上傳指定的資料夾，並將指定的標籤套用至所有新物件。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles" -Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [PutObject](#)。

Write-S3ObjectRetention

以下程式碼範例顯示如何使用 Write-S3ObjectRetention。

PowerShell V4 的工具

範例 1：命令會針對指定 S3 儲存貯體中的 'testfile.txt' 物件啟用控管保留模式，直到 2019 年 12 月 31 日 00:00:00 為止。

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [PutObjectRetention](#)。

使用 Tools for PowerShell V4 的 S3 Glacier 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 S3 Glacier 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執程式碼的指示。

主題

- [動作](#)

動作

Get-GLCJob

以下程式碼範例顯示如何使用 Get-GLCJob。

PowerShell V4 的工具

範例 1：傳回指定任務的詳細資訊。當任務成功完成時，Read-GCJobOutput cmdlet 可用來將任務的內容（封存或庫存清單）擷取至本機檔案系統。

```
Get-GLCJob -VaultName myvault -JobId "op1x...JSbthM"
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeJob](#)。

New-GLCVault

以下程式碼範例顯示如何使用 New-GLCVault。

PowerShell V4 的工具

範例 1：為使用者帳戶建立新的保存庫。由於未將值提供給 -AccountId 參數，Cmdlet 會使用預設值 "-" 來表示目前的帳戶。


```
New-GLCVault -VaultName myvault
```

輸出：

```
/01234567812/vaults/myvault
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateVault](#)。

Read-GLCJobOutput

以下程式碼範例顯示如何使用 Read-GLCJobOutput。

PowerShell V4 的工具

範例 1：下載排程在指定任務中擷取的封存內容，並將內容存放在磁碟上的檔案中。如果可用，下載會為您驗證檢查總和。如果需要，可以透過指定 來傳回包含檢查總和的整個回應-**Select '*'**。

```
Read-GLCJobOutput -VaultName myvault -JobId "HSWjArc...Zq2XLiW" -FilePath "c:\temp\blue.bin"
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetJobOutput](#)。

Start-GLCJob

以下程式碼範例顯示如何使用 Start-GLCJob。

PowerShell V4 的工具

範例 1：啟動任務，從使用者擁有的指定保存庫擷取封存。您可以使用 Get-GLCJob cmdlet 檢查任務的狀態。當任務成功完成時，Read-GCJobOutput cmdlet 可用來將封存的內容擷取至本機檔案系統。

```
Start-GLCJob -VaultName myvault -JobType "archive-retrieval" -JobDescription "archive retrieval" -ArchiveId "o909j...TX-TpIhQJw"
```

輸出：

JobId	JobOutputPath	Location
-----	-----	-----
op1x...JSbthM		/012345678912/vaults/test/jobs/op1xe...I4HqCHkSJSbthM

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [InitiateJob](#)。

Write-GLCArchive

以下程式碼範例顯示如何使用 Write-GLCArchive。

PowerShell V4 的工具

範例 1：將單一檔案上傳到指定的保存庫，並傳回封存 ID 和計算的檢查總和。

```
Write-GLCArchive -VaultName myvault -FilePath c:\temp\blue.bin
```

輸出：

FilePath	ArchiveId	Checksum
-----	-----	-----
C:\temp\blue.bin	o909jUUs...TTX-TpIhQJw	79f3e...f4395b

範例 2：將資料夾階層的內容上傳至使用者帳戶中指定的保存庫。對於每個上傳的檔案，Cmdlet 會發出檔案名稱、對應的封存 ID 和封存的計算檢查總和。

```
Write-GLCArchive -VaultName myvault -FolderPath . -Recurse
```

輸出：

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	-Ix3jlmU...iXiDh-XfOPA	7469e...3e86f1

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [UploadArchive](#)。

使用 Tools for PowerShell V4 的 Security Hub 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 Security Hub 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Get-SHUBFinding

以下程式碼範例顯示如何使用 Get-SHUBFinding。

PowerShell V4 的工具

範例 1：此命令會從 Amazon EC2；服務擷取 Security Hub 問題清單。

```
$filter = New-Object -TypeName Amazon.SecurityHub.Model.AwsSecurityFindingFilters
$filter.ResourceType = New-Object -TypeName Amazon.SecurityHub.Model.StringFilter -
Property @{
    Comparison = 'PREFIX'
    Value = 'AwsEc2'
}
Get-SHUBFinding -Filter $filter
```

範例 2：此命令會從 AWS 帳戶 ID 123456789012 擷取 Security Hub 問題清單。

```
$filter = New-Object -TypeName Amazon.SecurityHub.Model.AwsSecurityFindingFilters
$filter.AwsAccountId = New-Object -TypeName Amazon.SecurityHub.Model.StringFilter -
Property @{
```

```
Comparison = 'EQUALS'
Value = '123456789012'
}
Get-SHUBFinding -Filter $filter
```

範例 3：此命令會擷取標準 "pci-dss" 產生的 Security Hub 問題清單。

```
$filter = New-Object -TypeName Amazon.SecurityHub.Model.AwsSecurityFindingFilters
$filter.GeneratorId = New-Object -TypeName Amazon.SecurityHub.Model.StringFilter -
Property @{
    Comparison = 'PREFIX'
    Value = 'pci-dss'
}
Get-SHUBFinding -Filter $filter
```

範例 4：此命令會擷取工作流程狀態為 NOTIFIED 的 Security Hub 嚴重度問題清單。

```
$filter = New-Object -TypeName Amazon.SecurityHub.Model.AwsSecurityFindingFilters
$filter.SeverityLabel = New-Object -TypeName Amazon.SecurityHub.Model.StringFilter -
Property @{
    Comparison = 'EQUALS'
    Value = 'CRITICAL'
}
$filter.WorkflowStatus = New-Object -TypeName Amazon.SecurityHub.Model.StringFilter
-Property @{
    Comparison = 'EQUALS'
    Value = 'NOTIFIED'
}
Get-SHUBFinding -Filter $filter
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GetFindings](#)。

使用 Tools for PowerShell V4 的 Amazon SES 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 Amazon SES 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Get-SESIIdentity

以下程式碼範例顯示如何使用 Get-SESIIdentity。

PowerShell V4 的工具

範例 1：此命令會傳回包含特定 AWS 帳戶所有身分（電子郵件地址和網域）的清單，無論驗證狀態為何。

```
Get-SESIIdentity
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListIdentities](#)。

Get-SESSendQuota

以下程式碼範例顯示如何使用 Get-SESSendQuota。

PowerShell V4 的工具

範例 1：此命令會傳回使用者目前的傳送限制。

```
Get-SESSendQuota
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetSendQuota](#)。

Get-SESSendStatistic

以下程式碼範例顯示如何使用 Get-SESSendStatistic。

PowerShell V4 的工具

範例 1：此命令會傳回使用者的傳送統計資料。結果是資料點的清單，代表過去兩週的傳送活動。清單中的每個資料點都包含間隔 15 分鐘的統計資料。

Get-SESSendStatistic

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetSendStatistics](#)。

使用 Tools for PowerShell V4 的 Amazon SES API v2 範例 V4

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 Amazon SES API v2 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Send-SES2Email

以下程式碼範例顯示如何使用 Send-SES2Email。

PowerShell V4 的工具

範例 1：此範例示範如何傳送標準電子郵件訊息。

```
Send-SES2Email -FromEmailAddress "sender@example.com" -Destination_ToAddress  
"recipient@example.com" -Subject_Data "Email Subject" -Text_Data "Email Body"
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [SendEmail](#)。

使用 Tools for PowerShell V4 的 Amazon SNS 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 Amazon SNS 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Publish-SNSMessage

以下程式碼範例顯示如何使用 Publish-SNSMessage。

PowerShell V4 的工具

範例 1：此範例顯示使用單一 MessageAttribute 內嵌宣告來發佈訊息。

```
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -Message  
"Hello" -MessageAttribute  
@{'City'=[Amazon.SimpleNotificationService.Model.MessageAttributeValue]@{'DataType='String';  
StringValue ='AnyCity'}}
```

範例 2：此範例顯示預先宣告多個 MessageAttributes 的訊息發佈。

```
$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
```

- 如需 API 詳細資訊，請參閱在 AWS Tools for PowerShell Cmdlet 參考 (V4) 中 [發佈](#)。

使用 Tools for PowerShell V4 的 Amazon SQS 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 Amazon SQS 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Add-SQSPermission

以下程式碼範例顯示如何使用 Add-SQSPermission。

PowerShell V4 的工具

範例 1：此範例允許指定的 從指定的佇列 AWS 帳戶 傳送訊息。

```
Add-SQSPermission -Action SendMessage -AWSAccountId 80398EXAMPLE -Label  
SendMessagesFromMyQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/  
MyQueue
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [AddPermission](#)。

Clear-SQSQueue

以下程式碼範例顯示如何使用 Clear-SQSQueue。

PowerShell V4 的工具

範例 1：此範例會從指定的佇列刪除所有訊息。


```
Clear-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [PurgeQueue](#)。

Edit-SQSMessageVisibility

以下程式碼範例顯示如何使用 Edit-SQSMessageVisibility。

PowerShell V4 的工具

範例 1：此範例會將指定佇列中具有指定接收控點的訊息可見性逾時變更為 10 小時 (10 小時 * 60 分鐘 * 60 秒 = 36000 秒)。

```
Edit-SQSMessageVisibility -QueueUrl https://sqs.us-east-1.amazonaws.com/8039EXAMPLE/MyQueue -ReceiptHandle AQEBgGDh...J/Iqww== -VisibilityTimeout 36000
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ChangeMessageVisibility](#)。

Edit-SQSMessageVisibilityBatch

以下程式碼範例顯示如何使用 Edit-SQSMessageVisibilityBatch。

PowerShell V4 的工具

範例 1：此範例會變更 2 則訊息在指定佇列中具有指定接收控點的可見性逾時。第一個訊息的可見性逾時會變更為 10 小時 (10 小時 * 60 分鐘 * 60 秒 = 36000 秒)。第二個訊息的可見性逾時會變更為 5 小時 (5 小時 * 60 分鐘 * 60 秒 = 18000 秒)。

```
$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
```

輸出：

```
Failed      Successful
-----
{}          {Request2, Request1}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ChangeMessageVisibilityBatch](#)。

Get-SQSDeadLetterSourceQueue

以下程式碼範例顯示如何使用 Get-SQSDeadLetterSourceQueue。

PowerShell V4 的工具

範例 1：此範例列出依賴指定佇列做為無效字母佇列的任何佇列 URLs。

```
Get-SQSDeadLetterSourceQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

輸出：

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListDeadLetterSourceQueues](#)。

Get-SQSQueue

以下程式碼範例顯示如何使用 Get-SQSQueue。

PowerShell V4 的工具

範例 1：此範例列出所有佇列。

```
Get-SQSQueue
```

輸出：

```
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
```

範例 2：此範例列出以指定名稱開頭的任何佇列。

```
Get-SQSQueue -QueueNamePrefix My
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ListQueues](#)。

Get-SQSQueueAttribute

以下程式碼範例顯示如何使用 Get-SQSQueueAttribute。

PowerShell V4 的工具

範例 1：此範例列出指定佇列的所有屬性。

```
Get-SQSQueueAttribute -AttributeName All -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

輸出：

```
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                  : {[QueueArn, arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue], [ApproximateNumberOfMessages, 0],
                                [ApproximateNumberOfMessagesNotVisible, 0],
                                [ApproximateNumberOfMessagesDelayed, 0]...}

```

範例 2：此範例僅單獨列出指定佇列的指定屬性。

```

Get-SQSQueueAttribute -AttributeName MaximumMessageSize, VisibilityTimeout -QueueUrl
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue

```

輸出：

```

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]}

```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetQueueAttributes](#)。

Get-SQSQueueUrl

以下程式碼範例顯示如何使用 Get-SQSQueueUrl。

PowerShell V4 的工具

範例 1：此範例列出具有指定名稱的佇列 URL。

```
Get-SQSQueueUrl -QueueName MyQueue
```

輸出：

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetQueueUrl](#)。

New-SQSQueue

以下程式碼範例顯示如何使用 New-SQSQueue。

PowerShell V4 的工具

範例 1：此範例會建立具有指定名稱的佇列。

```
New-SQSQueue -QueueName MyQueue
```

輸出：

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateQueue](#)。

Receive-SQSMessage

以下程式碼範例顯示如何使用 Receive-SQSMessage。

PowerShell V4 的工具

範例 1：此範例列出指定佇列最多可收到接下來 10 則訊息的資訊。如果指定訊息屬性存在，該資訊將包含這些屬性的值。

```
Receive-SQSMessage -AttributeName SenderId, SentTimestamp -MessageAttributeName  
  StudentName, StudentGrade -MessageCount 10 -QueueUrl https://sqs.us-  
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

輸出：

```
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==
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ReceiveMessage](#)。

Remove-SQSMessage

以下程式碼範例顯示如何使用 Remove-SQSMessage。

PowerShell V4 的工具

範例 1：此範例會從指定的佇列刪除具有指定接收控點的訊息。

```
Remove-SQSMessage -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
-ReceiptHandle AQEBd329...v6gl8Q==
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteMessage](#)。

Remove-SQSMessageBatch

以下程式碼範例顯示如何使用 Remove-SQSMessageBatch。

PowerShell V4 的工具

範例 1：此範例會從指定的佇列刪除具有指定接收控點的 2 則訊息。

```
$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
```

輸出：

Failed	Successful
-----	-----
{}	{Request1, Request2}

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteMessageBatch](#)。

Remove-SQSPermission

以下程式碼範例顯示如何使用 Remove-SQSPermission。

PowerShell V4 的工具

範例 1：此範例會從指定的佇列移除具有指定標籤的許可設定。

```
Remove-SQSPermission -Label SendMessagesFromMyQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [RemovePermission](#)。

Remove-SQSQueue

以下程式碼範例顯示如何使用 Remove-SQSQueue。

PowerShell V4 的工具

範例 1：此範例會刪除指定的佇列。

```
Remove-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteQueue](#)。

Send-SQSMessage

以下程式碼範例顯示如何使用 Send-SQSMessage。

PowerShell V4 的工具

範例 1：此範例會將具有指定屬性和訊息內文的訊息傳送至指定的佇列，並延遲訊息傳遞 10 秒。

```
$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
```

輸出：

MD5ofMessageAttributes	MD5ofMessageBody	MessageId
-----	-----	-----
1d3e51347bc042efbdf6dda31EXAMPLE c739-4d0c-818b-1820eEXAMPLE	51b0a3256d59467f973009b73EXAMPLE	c35fed8f-

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [SendMessage](#)。

Send-SQSMessageBatch

以下程式碼範例顯示如何使用 Send-SQSMessageBatch。

PowerShell V4 的工具

範例 1：此範例會將 2 則具有指定屬性和訊息內文的訊息傳送至指定的佇列。第一個訊息的傳遞會延遲 15 秒，第二個訊息會延遲 10 秒。

```
$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

```

輸出：

```

Failed      Successful
-----
{}          {FirstMessage, SecondMessage}

```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [SendMessageBatch](#)。

Set-SQSQueueAttribute

以下程式碼範例顯示如何使用 Set-SQSQueueAttribute。

PowerShell V4 的工具

範例 1：此範例示範如何設定將佇列訂閱至 SNS 主題的政策。當訊息發佈至主題時，訊息會傳送至訂閱的佇列。

```
# 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 }
```

範例 2：此範例會設定指定佇列的指定屬性。

```
Set-SQSQueueAttribute -Attribute @{"DelaySeconds" = "10"; "MaximumMessageSize" =
  "131072"} -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [SetQueueAttributes](#)。

AWS STS 使用 Tools for PowerShell V4 的範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 來執行動作和實作常見案例 AWS STS。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Convert-STSAuthorizationMessage

以下程式碼範例顯示如何使用 Convert-STSAuthorizationMessage。

PowerShell V4 的工具

範例 1：解碼回應請求時傳回的所提供編碼訊息內容中包含的額外資訊。會對額外資訊進行編碼，因為授權狀態的詳細資訊可能構成請求該動作的使用者不應看到的特權資訊。

```
Convert-STSAuthorizationMessage -EncodedMessage "...encoded message..."
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DecodeAuthorizationMessage](#)。

Get-STSFederationToken

以下程式碼範例顯示如何使用 Get-STSFederationToken。

PowerShell V4 的工具

範例 1：使用 "Bob" 作為聯合身分使用者的名稱，請求有效期為一小時的聯合權杖。此名稱可用於參考資源型政策 (例如 Amazon S3 儲存貯體政策) 中的聯合身分使用者名稱。提供的 JSON 格式的 IAM 政策用於縮小 IAM 使用者可用的許可範圍。提供的政策授予的許可不得超過授予給請求使用者的許可，根據傳遞的政策和 IAM 使用者政策的交集，聯合身分使用者的最終許可是最具限制性的集合。

```
Get-STSFederationToken -Name "Bob" -Policy "...JSON policy..." -DurationInSeconds
3600
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetFederationToken](#)。

Get-STSSessionToken

以下程式碼範例顯示如何使用 Get-STSSessionToken。

PowerShell V4 的工具

範例 1：傳回 **Amazon.RuntimeAWSCredentials** 執行個體，其中包含有效期為設定時間段的臨時憑證。從目前 Shell 預設值中推斷出用於請求臨時憑證的憑證。若要指定其他憑證，請使用 -ProfileName 或 -AccessKey/-SecretKey 參數。

```
Get-STSSessionToken
```

輸出：

AccessKeyId	Expiration
SecretAccessKey	SessionToken
-----	-----
-----	-----
EXAMPLEACCESSKEYID	2/16/2015 9:12:28 PM
examplesecretaccesskey...	SamPleTokenN.....

範例 2：傳回 **Amazon.RuntimeAWSCredentials** 執行個體，其中包含有效期為一小時的臨時憑證。從指定的設定檔中取得用於提出請求的憑證。

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile
```

輸出：

AccessKeyId	Expiration
SecretAccessKey	SessionToken
-----	-----
-----	-----
EXAMPLEACCESSKEYID	2/16/2015 9:12:28 PM
examplesecretaccesskey...	SamPleTokenN.....

範例 3：使用在設定檔 'myprofilename' 中指定其憑證的帳戶關聯的 MFA 裝置識別碼和裝置提供的值，傳回 **Amazon.RuntimeAWSCredentials** 執行個體，其中包含有效期為一小時的臨時憑證。

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile -SerialNumber
YourMFADeviceSerialNumber -TokenCode 123456
```

輸出：

AccessKeyId	Expiration
SecretAccessKey	SessionToken
-----	-----
-----	-----
EXAMPLEACCESSKEYID	2/16/2015 9:12:28 PM
examplesecretaccesskey...	SamPleTokenN.....

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetSessionToken](#)。

Use-STSRole

以下程式碼範例顯示如何使用 Use-STSRole。

PowerShell V4 的工具

範例 1：傳回一組臨時登入資料（存取金鑰、私密金鑰和工作階段字符），可用於存取請求使用者通常無法存取 AWS 的資源一小時。傳回的憑證擁有所擔任角色之存取政策和提供的政策所允許的許可（您無法使用提供的政策來授予超過所擔任角色之存取政策所定義的許可）。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
Policy "...JSON policy..." -DurationInSeconds 3600
```

範例 2：傳回一組臨時憑證，有效期為一小時，這些憑證擁有所擔任角色之存取政策中定義的相同許可。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
DurationInSeconds 3600
```

範例 3：傳回一組臨時憑證，提供序號，以及與用於執行 cmdlet 之使用者憑證相關聯的 MFA 產生的權杖。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
DurationInSeconds 3600 -SerialNumber "GAHT12345678" -TokenCode "123456"
```

範例 4：傳回一組臨時憑證，這些臨時憑證擔任了客戶帳戶中定義的角色。對於第三方可以擔任的每個角色，客戶帳戶必須使用識別符來建立角色，必須在每次擔任角色時在 -ExternalId 參數中傳遞該識別符。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
DurationInSeconds 3600 -ExternalId "ABC123"
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [AssumeRole](#)。

Use-STSWebIdentityRole

以下程式碼範例顯示如何使用 Use-STSWebIdentityRole。

PowerShell V4 的工具

範例 1：針對已使用 Login with Amazon 身分提供程式進行身分驗證的使用者，傳回一組臨時憑證，有效期為一小時。憑證會擔任與角色 ARN 所識別角色相關聯的存取政策。或者，您可以將 JSON 政策傳遞至 -Policy 參數，以進一步改進存取許可 (您授予的許可不得超過與角色相關聯的許可中可用的許可)。提供給 -WebIdentityToken 的值是身分提供程式傳回的唯一使用者識別符。

```
Use-STSWebIdentityRole -DurationInSeconds 3600 -ProviderId "www.amazon.com"
-RoleSessionName "app1" -RoleArn "arn:aws:iam::123456789012:role/
FederatedWebIdentityRole" -WebIdentityToken "Atza...DVI0r1"
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [AssumeRoleWithWebIdentity](#)。

支援 使用 Tools for PowerShell V4 的範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 來執行動作和實作常見案例 支援。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執程式碼的指示。

主題

- [動作](#)

動作

Add-ASACommunicationToCase

以下程式碼範例顯示如何使用 Add-ASACommunicationToCase。

PowerShell V4 的工具

範例 1：將電子郵件通訊的內文新增至指定的案例。

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -  
CommunicationBody "Some text about the case"
```

範例 2：將電子郵件通訊的內文新增至指定的案例，以及電子郵件 CC 行中包含的一或多個電子郵件地址。

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -  
CcEmailAddress @"email1@address.com", "email2@address.com") -CommunicationBody  
"Some text about the case"
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [AddCommunicationToCase](#)。

Get-ASACase

以下程式碼範例顯示如何使用 Get-ASACase。

PowerShell V4 的工具

範例 1：傳回所有支援案例的詳細資訊。

```
Get-ASACase
```

範例 2：傳回自指定日期和時間以來所有支援案例的詳細資訊。

```
Get-ASACase -AfterTime "2013-09-10T03:06Z"
```

範例 3：傳回前 10 個支援案例的詳細資訊，包括已解決的案例。

```
Get-ASACase -MaxResult 10 -IncludeResolvedCases $true
```

範例 4：傳回單一指定支援案例的詳細資訊。

```
Get-ASACase -CaseIdList "case-12345678910-2013-c4c1d2bf33c5cf47"
```

範例 5：傳回指定支援案例的詳細資訊。

```
Get-ASACase -CaseIdList @("case-12345678910-2013-c4c1d2bf33c5cf47",  
"case-18929034710-2011-c4fdeabf33c5cf47")
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeCases](#)。

Get-ASACommunication

以下程式碼範例顯示如何使用 Get-ASACommunication。

PowerShell V4 的工具

範例 1：傳回指定案例的所有通訊。

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

範例 2：針對指定的案例，傳回自 2012 年 1 月 1 日 UTC 午夜以來的所有通訊。

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -AfterTime  
"2012-01-10T00:00Z"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeCommunications](#)。

Get-ASAService

以下程式碼範例顯示如何使用 Get-ASAService。

PowerShell V4 的工具

範例 1：傳回所有可用的服務代碼、名稱和類別。

```
Get-ASAService
```

範例 2：使用指定的程式碼傳回服務的名稱和類別。

```
Get-ASAService -ServiceCodeList "amazon-cloudfront"
```

範例 3：傳回指定服務代碼的名稱和類別。

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch")
```

範例 4：傳回指定服務代碼的名稱和類別（日文）。目前支援英文 ("en") 和日文 ("ja") 語言代碼。

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch") -  
Language "ja"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeServices](#)。

Get-ASASeverityLevel

以下程式碼範例顯示如何使用 Get-ASASeverityLevel。

PowerShell V4 的工具

範例 1：傳回可指派給 AWS 支援案例的嚴重性等級清單。

```
Get-ASASeverityLevel
```

範例 2：傳回可指派給 AWS 支援案例的嚴重性等級清單。關卡的名稱會以日文傳回。

```
Get-ASASeverityLevel -Language "ja"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeSeverityLevels](#)。

Get-ASATrustedAdvisorCheck

以下程式碼範例顯示如何使用 Get-ASATrustedAdvisorCheck。

PowerShell V4 的工具

範例 1：傳回 Trusted Advisor 檢查的集合。您必須指定語言參數，可接受英文輸出的「en」或日文輸出的「ja」。

```
Get-ASATrustedAdvisorCheck -Language "en"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeTrustedAdvisorChecks](#)。

Get-ASATrustedAdvisorCheckRefreshStatus

以下程式碼範例顯示如何使用 Get-ASATrustedAdvisorCheckRefreshStatus。

PowerShell V4 的工具

範例 1：傳回指定檢查的重新整理請求目前狀態。Request-ASATrustedAdvisorCheckRefresh 可用來請求重新整理檢查的狀態資訊。

```
Get-ASATrustedAdvisorCheckRefreshStatus -CheckId @("checkid1", "checkid2")
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeTrustedAdvisorCheckRefreshStatuses](#)。

Get-ASATrustedAdvisorCheckResult

以下程式碼範例顯示如何使用 Get-ASATrustedAdvisorCheckResult。

PowerShell V4 的工具

範例 1：傳回 Trusted Advisor 檢查的結果。您可以使用 `Get-ASATrustedAdvisorChecks` 取得可用的 Trusted Advisor 檢查清單。`Get-ASATrustedAdvisorChecks` 輸出是檢查的整體狀態、上次執行檢查的時間戳記，以及特定檢查的唯一檢查 ID。若要讓結果輸出以日文顯示，請新增 `-Language "ja"` 參數。

```
Get-ASATrustedAdvisorCheckResult -CheckId "checkid1"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeTrustedAdvisorCheckResult](#)。

Get-ASATrustedAdvisorCheckSummary

以下程式碼範例顯示如何使用 `Get-ASATrustedAdvisorCheckSummary`。

PowerShell V4 的工具

範例 1：傳回指定 Trusted Advisor 檢查的最新摘要。

```
Get-ASATrustedAdvisorCheckSummary -CheckId "checkid1"
```

範例 2：傳回指定 Trusted Advisor 檢查的最新摘要。

```
Get-ASATrustedAdvisorCheckSummary -CheckId @("checkid1", "checkid2")
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeTrustedAdvisorCheckSummaries](#)。

New-ASACase

以下程式碼範例顯示如何使用 `New-ASACase`。

PowerShell V4 的工具

範例 1：在 AWS 支援中心建立新案例。您可以使用 `Get-ASAService` cmdlet 取得 `-ServiceCode` 和 `-CategoryCode` 參數的值。`-SeverityCode` 參數的值可使用 `Get-ASASeverityLevel` cmdlet 取得。`-IssueType` 參數值可以是「customer-service」或「technical」。如果成功，則會輸出 AWS 支援案例編號。根據預設，案例將以英文處理，以使用日文新增 `-Language "ja"` 參數。`-ServiceCode`、`-CategoryCode`、`-Subject` 和 `-CommunicationBody` 參數是強制性的。

```
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"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateCase](#)。

Request-ASATrustedAdvisorCheckRefresh

以下程式碼範例顯示如何使用 Request-ASATrustedAdvisorCheckRefresh。

PowerShell V4 的工具

範例 1：請求重新整理指定的 Trusted Advisor 檢查。

```
Request-ASATrustedAdvisorCheckRefresh -CheckId "checkid1"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [RefreshTrustedAdvisorCheck](#)。

Resolve-ASACase

以下程式碼範例顯示如何使用 Resolve-ASACase。

PowerShell V4 的工具

範例 1：傳回指定案例的初始狀態，以及在呼叫解決它之後的目前狀態。

```
Resolve-ASACase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ResolveCase](#)。

使用 Tools for PowerShell V4 的 Systems Manager 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 Systems Manager 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

Add-SSMResourceTag

以下程式碼範例顯示如何使用 Add-SSMResourceTag。

PowerShell V4 的工具

範例 1：此範例示範使用新標籤更新維護時段。如果命令成功，則無輸出訊息。此範例使用的語法需要 PowerShell 版本 3 或更新版本。

```
$option1 = @{Key="Stack";Value=@"Production"}  
Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType  
"MaintenanceWindow" -Tag $option1
```

範例 2：使用 PowerShell 版本 2 時，必須使用 New-Object 來建立每個標籤。如果命令成功，則無輸出訊息。

```
$tag1 = New-Object Amazon.SimpleSystemsManagement.Model.Tag  
$tag1.Key = "Stack"  
$tag1.Value = "Production"  
  
Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType  
"MaintenanceWindow" -Tag $tag1
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [AddTagsToResource](#)。

Edit-SSMDocumentPermission

以下程式碼範例顯示如何使用 Edit-SSMDocumentPermission。

PowerShell V4 的工具

範例 1：此範例將文件的「共用」許可新增至所有帳戶。如果命令成功，則無輸出訊息。

```
Edit-SSMDocumentPermission -Name "RunShellScript" -PermissionType "Share" -
AccountIdsToAdd all
```

範例 2：此範例將文件的「共用」許可新增至特定帳戶。如果命令成功，則無輸出訊息。

```
Edit-SSMDocumentPermission -Name "RunShellScriptNew" -PermissionType "Share" -
AccountIdsToAdd "123456789012"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ModifyDocumentPermission](#)。

Get-SSMActivation

以下程式碼範例顯示如何使用 Get-SSMActivation。

PowerShell V4 的工具

範例 1：此範例提供有關您帳戶中啟用的詳細資訊。

```
Get-SSMActivation
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeActivations](#)。

Get-SSMAssociation

以下程式碼範例顯示如何使用 Get-SSMAssociation。

PowerShell V4 的工具

範例 1：此範例說明執行個體與文件之間的關聯。

```
Get-SSMAssociation -InstanceId "i-0000293ffd8c57862" -Name "AWS-UpdateSSMAgent"
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeAssociation](#)。

Get-SSMAssociationExecution

以下程式碼範例顯示如何使用 Get-SSMAssociationExecution。

PowerShell V4 的工具

範例 1：此範例傳回所提供關聯 ID 的執行

```
Get-SSMAssociationExecution -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

輸出：

```
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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeAssociationExecutions](#)。

Get-SSMAssociationExecutionTarget

以下程式碼範例顯示如何使用 Get-SSMAssociationExecutionTarget。

PowerShell V4 的工具

範例 1：此範例顯示資源 ID 及其執行狀態，它們是關聯執行目標的一部分

```
Get-SSMAssociationExecutionTarget -AssociationId 123a45a0-
c678-9012-3456-78901234db5e -ExecutionId 123a45a0-c678-9012-3456-78901234db5e |
Select-Object ResourceId, Status
```

輸出：

ResourceId	Status
-----	-----
i-0b1b2a3456f7a890b	Success
i-01c12a45d6fc7a89f	Success
i-0a1caf234f56d7dc8	Success
i-012a3fd45af6dbcfe	Failed
i-0ddc1df23c4a5fb67	Success

範例 2：此命令會檢查自昨天以來特定自動化的特定執行，其中關聯有命令文件。它會進一步檢查關聯執行是否失敗，如果是，則會顯示該執行的命令調用詳細資訊以及執行個體 ID

```
$AssociationExecution= Get-SSMAssociationExecutionTarget -
AssociationId 1c234567-890f-1aca-a234-5a678d901cb0 -ExecutionId
12345ca12-3456-2345-2b45-23456789012 |
Where-Object {$_.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
```

```
}
}
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeAssociationExecutionTargets](#)。

Get-SSMAssociationList

以下程式碼範例顯示如何使用 Get-SSMAssociationList。

PowerShell V4 的工具

範例 1：此範例列出了執行個體的所有關聯。此範例使用的語法需要 PowerShell 版本 3 或更新版本。

```
$filter1 = @{"Key"="InstanceId";Value=@"i-0000293ffd8c57862"}
Get-SSMAssociationList -AssociationFilterList $filter1
```

輸出：

```
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}

```

範例 2：此範例列出了組態文件的所有關聯。此範例使用的語法需要 PowerShell 版本 3 或更新版本。

```

$filter2 = @{"Key"="Name";Value=@"AWS-UpdateSSMAgent"}
Get-SSMAssociationList -AssociationFilterList $filter2

```

輸出：

```

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}

```

範例 3：使用 PowerShell 版本 2 時，必須使用 New-Object 來建立每個篩選條件。

```

$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.AssociationFilter
$filter1.Key = "InstanceId"
$filter1.Value = "i-0000293ffd8c57862"

Get-SSMAssociationList -AssociationFilterList $filter1

```

輸出：

```

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}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListAssociations](#)。

Get-SSMAssociationVersionList

以下程式碼範例顯示如何使用 Get-SSMAssociationVersionList。

PowerShell V4 的工具

範例 1：此範例示範擷取所提供之關聯的所有版本。

```
Get-SSMAssociationVersionList -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

輸出：

```
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}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListAssociationVersions](#)。

Get-SSMAutomationExecution

以下程式碼範例顯示如何使用 Get-SSMAutomationExecution。

PowerShell V4 的工具

範例 1：此範例顯示自動化執行的詳細資訊。

```
Get-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"
```

輸出：

```
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_V2QyyN7NhSZQYpmVzpEc4oSnj2GLTNYnXUHsTbqJkNMoDgubmbtthLmZyaiUYek0RIrA42-
                           fv1x-04q5Fjff6g1h
                           Yb6TI5b0GQeeNrpwNvpDzm0-
PSR1swlAbg9fdM9BcNjyrznspUkWpuKu9EC10u6v30XU1KC9nZ7mPlWMFZNkSioQqpWWEvMw-
GZktsQzm67q0hUhBN0LWYhbS
                           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...}

```

範例 2：此範例列出了指定自動化執行 ID 的步驟詳細資訊

```

Get-SSMAutomationExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object -ExpandProperty StepExecutions | Select-Object
StepName, Action, StepStatus, ValidNextSteps

```

輸出：

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	{}

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetAutomationExecution](#)。

Get-SSMAutomationExecutionList

以下程式碼範例顯示如何使用 Get-SSMAutomationExecutionList。

PowerShell V4 的工具

範例 1：此範例說明與帳戶關聯的所有作用中和已終止的自動化執行。

```
Get-SSMAutomationExecutionList
```

輸出：

```
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]]}
```

範例 2：此範例顯示具有 AutomationExecutionStatus 而非 "Success" 之執行的 ExecutionID、文件、執行開始/結束時間戳記

```
Get-SSMAutomationExecutionList | Where-Object AutomationExecutionStatus
    -ne "Success" | Select-Object AutomationExecutionId, DocumentName,
    AutomationExecutionStatus, ExecutionStartTime, ExecutionEndTime | Format-Table -
    AutoSize
```

輸出：

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

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeAutomationExecutions](#)。

Get-SSMAutomationStepExecution

以下程式碼範例顯示如何使用 Get-SSMAutomationStepExecution。

PowerShell V4 的工具

範例 1：此範例顯示有關自動化工作流程中所有作用中和已終止之步驟執行的資訊。

```
Get-SSMAutomationStepExecution -AutomationExecutionId e1d2bad3-4567-8901-ae23-456c7c8901be | Select-Object StepName, Action, StepStatus
```

輸出：

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

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeAutomationStepExecutions](#)。

Get-SSMAvailablePatch

以下程式碼範例顯示如何使用 Get-SSMAvailablePatch。

PowerShell V4 的工具

範例 1：此範例示範取得適用於 Windows Server 2012 的所有可用修補程式，這些修補程式的 MSRC 嚴重性為「關鍵」。此範例使用的語法需要 PowerShell 版本 3 或更新版本。


```
$filter1 = @{"Key"="PRODUCT";Values=@("WindowsServer2012")}  
$filter2 = @{"Key"="MSRC_SEVERITY";Values=@("Critical")}  
  
Get-SSMAvailablePatch -Filter $filter1,$filter2
```

輸出：

```
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  
...
```

範例 2：使用 PowerShell 版本 2 時，必須使用 New-Object 來建立每個篩選條件。

```
$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
```

範例 3：此範例示範擷取過去 20 天內發行並適用於符合 WindowsServer2019 的產品的所有更新

```
Get-SSMAvailablePatch | Where-Object ReleaseDate -ge (Get-Date).AddDays(-20) |
Where-Object Product -eq "WindowsServer2019" | Select-Object ReleaseDate, Product,
Title
```

輸出：

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)

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeAvailablePatches](#)。

Get-SSMCommand

以下程式碼範例顯示如何使用 Get-SSMCommand。

PowerShell V4 的工具

範例 1：此範例列出了所請求的所有命令。

```
Get-SSMCommand
```

輸出：

```
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           : {}

```

範例 2：此範例示範取得特定命令的狀態。

```
Get-SSMCommand -CommandId "4b75a163-d39a-4d97-87c9-98ae52c6be35"
```

範例 3：此範例示範擷取在 2019-04-01T00:00:00Z 之後調用的所有 SSM 命令

```

Get-SSMCommand -Filter @{Key="InvokedAfter";Value="2019-04-01T00:00:00Z"} | Select-
Object CommandId, DocumentName, Status, RequestedDateTime | Sort-Object -Property
RequestedDateTime -Descending

```

輸出：

CommandId RequestedDateTime ----- -----	DocumentName	Status	
edb1b23e-456a-7adb-aef8-90e-012ac34f 5:45:23 AM	AWS-RunPowerShellScript	Cancelled	4/16/2019
1a2dc3fb-4567-890d-a1ad-234b5d6bc7d9 9:19:42 AM	AWS-ConfigureAWSPackage	Success	4/6/2019
12c3456c-7e90-4f12-1232-1234f5b67893 4:13:07 AM	KT-Retrieve-Cloud-Type-Win	Failed	4/2/2019
fe123b45-240c-4123-a2b3-234bdd567ecf 2:27:31 PM	AWS-RunInspecChecks	Failed	4/1/2019
1eb23aa4-567d-4123-12a3-4c1c2ab34561 1:05:55 PM	AWS-RunPowerShellScript	Success	4/1/2019
1c2f3bb4-ee12-4bc1-1a23-12345eea123e 11:13:09 AM	AWS-RunInspecChecks	Failed	4/1/2019

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListCommands](#)。

Get-SSMCommandInvocation

以下程式碼範例顯示如何使用 Get-SSMCommandInvocation。

PowerShell V4 的工具

範例 1：此範例列出了命令的所有調用。

```
Get-SSMCommandInvocation -CommandId "b8eac879-0541-439d-94ec-47a80d554f44" -Detail $true
```

輸出：

```
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         :
```

範例 2：此範例列出了用於調用命令 ID 為 e1eb2e3c-ed4c-5123-45c1-234f5612345f 的 CommandPlugins

```
Get-SSMCommandInvocation -CommandId e1eb2e3c-ed4c-5123-45c1-234f5612345f -Detail: $true | Select-Object -ExpandProperty CommandPlugins
```

輸出：

```
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

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListCommandInvocations](#)。

Get-SSMCommandInvocationDetail

以下程式碼範例顯示如何使用 Get-SSMCommandInvocationDetail。

PowerShell V4 的工具

範例 1：此範例顯示執行個體上所執行之命令的詳細資訊。

```
Get-SSMCommandInvocationDetail -InstanceId "i-0cb2b964d3e14fd9f" -CommandId
"b8eac879-0541-439d-94ec-47a80d554f44"
```

輸出：

```

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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetCommandInvocation](#)。

Get-SSMComplianceItemList

以下程式碼範例顯示如何使用 Get-SSMComplianceItemList。

PowerShell V4 的工具

範例 1：此範例列出了指定資源 ID 和類型的合規項目清單，篩選合規類型為「關聯」

```
Get-SSMComplianceItemList -ResourceId i-1a2caf345f67d0dc2 -ResourceType
ManagedInstance -Filter @{"Key"="ComplianceType";Values="Association"}
```

輸出：

```
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            :
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListComplianceItems](#)。

Get-SSMComplianceSummaryList

以下程式碼範例顯示如何使用 Get-SSMComplianceSummaryList。

PowerShell V4 的工具

範例 1：此範例傳回所有合規類型的合規與不合規資源計數摘要。

```
Get-SSMComplianceSummaryList
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [ListComplianceSummaries](#)。

Get-SSMConnectionStatus

以下程式碼範例顯示如何使用 Get-SSMConnectionStatus。

PowerShell V4 的工具

範例 1：此範例示範擷取執行個體的 Session Manager 連線狀態，以判斷執行個體是否已連線並準備好接收 Session Manager 連線。

```
Get-SSMConnectionStatus -Target i-0a1caf234f12d3dc4
```

輸出：

```
Status      Target
-----
Connected i-0a1caf234f12d3dc4
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetConnectionStatus](#)。

Get-SSMDefaultPatchBaseline

以下程式碼範例顯示如何使用 Get-SSMDefaultPatchBaseline。

PowerShell V4 的工具

範例 1：此範例顯示預設修補基準。

```
Get-SSMDefaultPatchBaseline
```

輸出：

```
arn:aws:ssm:us-west-2:123456789012:patchbaseline/pb-04fb4ae6142167966
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [GetDefaultPatchBaseline](#)。

Get-SSMDeployablePatchSnapshotForInstance

以下程式碼範例顯示如何使用 Get-SSMDeployablePatchSnapshotForInstance。

PowerShell V4 的工具

範例 1：此範例顯示執行個體所用之修補基準的目前快照。此命令必須使用執行個體憑證從執行個體執行。為確保它使用執行個體憑證，範例會將 **Amazon.Runtime.InstanceProfileAWSCredentials** 物件傳遞至憑證參數。

```
$credentials = [Amazon.Runtime.InstanceProfileAWSCredentials]::new()
Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f" -Credentials $credentials
```

輸出：

```
InstanceId          SnapshotDownloadUrl
-----
i-0cb2b964d3e14fd9f https://patch-baseline-snapshot-us-west-2.s3-us-west-2.amazonaws.com/853d0d3db0f0cafe...1692/4681775b-098f-4435...
```


範例 2：此範例顯示如何取得完整的 `SnapshotDownloadUrl`。此命令必須使用執行個體憑證從執行個體執行。為確保它使用執行個體憑證，此範例會將 PowerShell 工作階段設定為使用 **Amazon.Runtime.InstanceProfileAWSCredentials** 物件。

```
Set-AWSCredential -Credential  
([Amazon.Runtime.InstanceProfileAWSCredentials]::new())  
(Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-  
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f").SnapshotDownloadUrl
```

輸出：

```
https://patch-baseline-snapshot-us-west-2.s3-us-  
west-2.amazonaws.com/853d0d3db0f0cafe...
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetDeployablePatchSnapshotForInstance](#)。

Get-SSMDocument

以下程式碼範例顯示如何使用 `Get-SSMDocument`。

PowerShell V4 的工具

範例 1：此範例傳回文件的內容。

```
Get-SSMDocument -Name "RunShellScript"
```

輸出：

```
Content  
-----  
{...
```

範例 2：此範例顯示文件的完整內容。

```
(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 }}"
        }
      }
    ]
  }
}

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GetDocument](#)。

Get-SSMDocumentDescription

以下程式碼範例顯示如何使用 Get-SSMDocumentDescription。

PowerShell V4 的工具

範例 1：此範例傳回文件的相關資訊。

```
Get-SSMDocumentDescription -Name "RunShellScript"
```

輸出：

```

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

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeDocument](#)。

Get-SSMDocumentList

以下程式碼範例顯示如何使用 Get-SSMDocumentList。

PowerShell V4 的工具

範例 1：列出帳戶中的所有組態文件。

```
Get-SSMDocumentList
```

輸出：

```

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
...
```

範例 2：此範例示範擷取名稱符合 "Platform" 的所有自動化文件

```
Get-SSMDocumentList -DocumentFilterList @{Key="DocumentType";Value="Automation"} |
Where-Object Name -Match "Platform"
```

輸出：

```
DocumentFormat    : JSON
DocumentType      : Automation
DocumentVersion   : 7
Name              : KT-Get-Platform
Owner            : 987654123456
PlatformTypes     : {Windows, Linux}
SchemaVersion     : 0.3
Tags              : {}
TargetType        :
VersionName       :
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListDocuments](#)。

Get-SSMDocumentPermission

以下程式碼範例顯示如何使用 Get-SSMDocumentPermission。

PowerShell V4 的工具

範例 1：此範例列出了文件的所有版本。

```
Get-SSMDocumentVersionList -Name "RunShellScript"
```

輸出：

CreatedDate	DocumentVersion	IsDefaultVersion	Name
-----	-----	-----	----
2/24/2017 5:25:13 AM	1	True	RunShellScript

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeDocumentPermission](#)。

Get-SSMDocumentVersionList

以下程式碼範例顯示如何使用 Get-SSMDocumentVersionList。

PowerShell V4 的工具

範例 1：此範例列出了文件的所有版本。

```
Get-SSMDocumentVersionList -Name "AWS-UpdateSSMAgent"
```

輸出：

```

CreatedDate      : 6/1/2021 5:19:10 PM
DocumentFormat   : JSON
DocumentVersion  : 1
IsDefaultVersion : True
Name             : AWS-UpdateSSMAgent
Status          : Active

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListDocumentVersions](#)。

Get-SSMEffectiveInstanceAssociationList

以下程式碼範例顯示如何使用 Get-SSMEffectiveInstanceAssociationList。

PowerShell V4 的工具

範例 1：此範例說明執行個體的有效關聯。

```
Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -MaxResult 5
```

輸出：

AssociationId	Content
-----	-----
d8617c07-2079-4c18-9847-1655fc2698b0	{...

範例 2：此範例顯示執行個體有效關聯的內容。

```
(Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -
MaxResult 5).Content
```

輸出：

```
{
  "schemaVersion": "1.2",
  "description": "Update the Amazon SSM Agent to the latest version or specified
version.",
  "parameters": {
    "version": {
      "default": "",
      "description": "(Optional) A specific version of the Amazon SSM Agent to
install. If not specified, the agen
t will be updated to the latest version.",
      "type": "String"
    },
    "allowDowngrade": {
      "default": "false",
      "description": "(Optional) Allow the Amazon SSM Agent service to be
downgraded to an earlier version. If set
to false, the service can be upgraded to newer versions only (default). If set to
true, specify the earlier version.",
      "type": "String",
      "allowedValues": [
        "true",
        "false"
      ]
    }
  },
  "runtimeConfig": {
    "aws:updateSsmAgent": {
      "properties": [
        {
```

```

        "agentName": "amazon-ssm-agent",
        "source": "https://s3.{Region}.amazonaws.com/amazon-ssm-{Region}/
ssm-agent-manifest.json",
        "allowDowngrade": "{{ allowDowngrade }}",
        "targetVersion": "{{ version }}"
    }
  ]
}
}
}

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeEffectiveInstanceAssociations](#)。

Get-SSMEffectivePatchesForPatchBaseline

以下程式碼範例顯示如何使用 Get-SSMEffectivePatchesForPatchBaseline。

PowerShell V4 的工具

範例 1：此範例列出了所有修補基準，結果清單上限為 1。

```
Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1
```

輸出：

Patch	PatchStatus
-----	-----
Amazon.SimpleSystemsManagement.Model.Patch	
Amazon.SimpleSystemsManagement.Model.PatchStatus	

範例 2：此範例顯示所有修補基準的修補程式狀態，結果清單上限為 1。

```
(Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1).PatchStatus
```

輸出：

ApprovalDate	DeploymentStatus
-----	-----

12/21/2010 6:00:00 PM APPROVED

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeEffectivePatchesForPatchBaseline](#)。

Get-SSMInstanceAssociationsStatus

以下程式碼範例顯示如何使用 Get-SSMInstanceAssociationsStatus。

PowerShell V4 的工具

範例 1：此範例顯示執行個體關聯的詳細資訊。

```
Get-SSMInstanceAssociationsStatus -InstanceId "i-0000293ffd8c57862"
```

輸出：

```
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
```

範例 2：此範例檢查指定執行個體 ID 的執行個體關聯狀態，並進一步顯示這些關聯的執行狀態

```
Get-SSMInstanceAssociationsStatus -InstanceId i-012e3cb4df567e8aa | ForEach-Object
{Get-SSMAssociationExecution -AssociationId .AssociationId}
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeInstanceAssociationsStatus](#)。

Get-SSMInstanceInformation

以下程式碼範例顯示如何使用 Get-SSMInstanceInformation。

PowerShell V4 的工具

範例 1：此範例顯示每個執行個體的詳細資訊。

```
Get-SSMInstanceInformation
```

輸出：

```
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
```

範例 2：此範例示範如何使用 -Filter 參數，僅篩選 **us-east-1** 區域中具有 **AgentVersion** 之 的 AWS Systems Manager 執行個體的結果 **2.2.800.0**。InstanceInformation API 參考主題中提供

了有效的 -Filter 鍵值清單 (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
```

輸出：

```
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

```

範例 3：此範例示範如何使用 `-InstanceInformationFilterList` 參數，將結果篩選為 **Windows** 或 區域中 **us-east-1PlatformTypes** 的 AWS Systems Manager 執行個體 **Linux**。InstanceInformation API 參考主題中提供了有效的 `-InstanceInformationFilterList` 鍵值清單 (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

```

輸出：

```

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

```

範例 4：此範例列出了 ssm 受管執行個體，並將 InstanceId、PingStatus、LastPingDateTime 和 PlatformName 匯出至 csv 檔案。

```

Get-SSMInstanceInformation | Select-Object InstanceId, PingStatus, LastPingDateTime,
PlatformName | Export-Csv Instance-details.csv -NoTypeInfoInformation

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeInstanceInformation](#)。

Get-SSMInstancePatch

以下程式碼範例顯示如何使用 Get-SSMInstancePatch。

PowerShell V4 的工具

範例 1：此範例示範取得執行個體的修補程式合規性詳細資訊。

```

Get-SSMInstancePatch -InstanceId "i-08ee91c0b17045407"

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeInstancePatches](#)。

Get-SSMInstancePatchState

以下程式碼範例顯示如何使用 Get-SSMInstancePatchState。

PowerShell V4 的工具

範例 1：此範例示範取得一個執行個體的修補程式摘要狀態。

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407"
```

範例 2：此範例示範取得兩個執行個體的修補程式摘要狀態。

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407","i-09a618aec652973a9"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeInstancePatchStates](#)。

Get-SSMInstancePatchStatesForPatchGroup

以下程式碼範例顯示如何使用 Get-SSMInstancePatchStatesForPatchGroup。

PowerShell V4 的工具

範例 1：此範例示範取得修補程式群組中每個執行個體的修補程式摘要狀態。

```
Get-SSMInstancePatchStatesForPatchGroup -PatchGroup "Production"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeInstancePatchStatesForPatchGroup](#)。

Get-SSMInventory

以下程式碼範例顯示如何使用 Get-SSMInventory。

PowerShell V4 的工具

範例 1：此範例示範取得庫存的自訂中繼資料。

Get-SSMInventory

輸出：

```
Data
  Id
----
--
{[AWS:InstanceInformation,
 Amazon.SimpleSystemsManagement.Model.InventoryResultItem]} i-0cb2b964d3e14fd9f
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [GetInventory](#)。

Get-SSMInventoryEntriesList

以下程式碼範例顯示如何使用 Get-SSMInventoryEntriesList。

PowerShell V4 的工具

範例 1：此範例列出了執行個體的所有自訂庫存項目。

```
Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
"Custom:RackInfo"
```

輸出：

```
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
```

範例 2：此範例列出了詳細資訊。

```
(Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
"Custom:RackInfo").Entries
```

輸出：

Key	Value
---	-----
RackLocation	Bay B/Row C/Rack D/Shelf E

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListInventoryEntries](#)。

Get-SSMInventoryEntryList

以下程式碼範例顯示如何使用 Get-SSMInventoryEntryList。

PowerShell V4 的工具

範例 1：此範例會擷取執行個體的 **AWS:Network** 類型庫存項目。

```
Get-SSMInventoryEntryList -InstanceId mi-088dcb0ecea37b076 -TypeName AWS:Network |
Select-Object -ExpandProperty Entries
```

輸出：

Key	Value
---	-----
DHCPServer	172.31.11.2
DNSServer	172.31.0.1
Gateway	172.31.11.2
IPv4	172.31.11.222
IPv6	fe12::3456:7da8:901a:12a3
MacAddress	1A:23:4E:5B:FB:67
Name	Amazon Elastic Network Adapter
SubnetMask	255.255.240.0

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [Get-SSMInventoryEntryList](#)。

Get-SSMInventorySchema

以下程式碼範例顯示如何使用 Get-SSMInventorySchema。

PowerShell V4 的工具

範例 1：此範例傳回帳戶的庫存類型名稱清單。

```
Get-SSMInventorySchema
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetInventorySchema](#)。

Get-SSMLatestEC2Image

以下程式碼範例顯示如何使用 Get-SSMLatestEC2Image。

PowerShell V4 的工具

範例 1：此範例列出所有最新的 Windows AMIs。

```
PS Get-SSMLatestEC2Image -Path ami-windows-latest
```

輸出：

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	
...	

範例 2：此範例會擷取 us-west-2 區域特定 Amazon Linux 映像的 AMI ID。

```
PS Get-SSMLatestEC2Image -Path ami-amazon-linux-latest -ImageName amzn-ami-hvm-  
x86_64-ebs -Region us-west-2
```

輸出：

```
ami-09b92cd132204c704
```

範例 3：此範例列出符合指定萬用字元表達式的所有最後 Windows AMIs。


```
Get-SSMLatestEC2Image -Path ami-windows-latest -ImageName *Windows*2019*English*
```

輸出：

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
...	

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [Get-SSMLatestEC2Image](#)。

Get-SSMMaintenanceWindow

以下程式碼範例顯示如何使用 Get-SSMMaintenanceWindow。

PowerShell V4 的工具

範例 1：此範例示範取得有關維護時段的詳細資訊。

```
Get-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d"
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetMaintenanceWindow](#)。

Get-SSMMaintenanceWindowExecution

以下程式碼範例顯示如何使用 Get-SSMMaintenanceWindowExecution。

PowerShell V4 的工具

範例 1：此範例列出了作為維護時段執行的一部分執行之任務的資訊。

```
Get-SSMMaintenanceWindowExecution -WindowExecutionId "518d5565-5969-4cca-8f0e-da3b2a638355"
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetMaintenanceWindowExecution](#)。

Get-SSMMaintenanceWindowExecutionList

以下程式碼範例顯示如何使用 Get-SSMMaintenanceWindowExecutionList。

PowerShell V4 的工具

範例 1：此範例列出了維護時段的所有執行。

```
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d"
```

輸出：

```
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
```

範例 2：此範例列出了指定日期之前維護時段的所有執行。

```
$option1 = @{Key="ExecutedBefore";Values=@"2016-11-04T05:00:00Z"}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
$option1
```

範例 3：此範例列出了指定日期之後維護時段的所有執行。

```
$option1 = @{Key="ExecutedAfter";Values=@"2016-11-04T05:00:00Z"}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
$option1
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeMaintenanceWindowExecutions](#)。

Get-SSMMaintenanceWindowExecutionTask

以下程式碼範例顯示如何使用 Get-SSMMaintenanceWindowExecutionTask。

PowerShell V4 的工具

範例 1：此範例列出了有關屬於維護時段執行之任務的資訊。

```
Get-SSMMaintenanceWindowExecutionTask -TaskId "ac0c6ae1-daa3-4a89-832e-d384503b6586"
-WindowExecutionId "518d5565-5969-4cca-8f0e-da3b2a638355"
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [GetMaintenanceWindowExecutionTask](#)。

Get-SSMMaintenanceWindowExecutionTaskInvocationList

以下程式碼範例顯示如何使用 Get-SSMMaintenanceWindowExecutionTaskInvocationList。

PowerShell V4 的工具

範例 1：此範例列出了作為維護時段執行的一部分執行之任務的調用。

```
Get-SSMMaintenanceWindowExecutionTaskInvocationList -TaskId "ac0c6ae1-
daa3-4a89-832e-d384503b6586" -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"
```

輸出：

```
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 :
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeMaintenanceWindowExecutionTaskInvocations](#)。

Get-SSMMaintenanceWindowExecutionTaskList

以下程式碼範例顯示如何使用 Get-SSMMaintenanceWindowExecutionTaskList。

PowerShell V4 的工具

範例 1：此範例列出了與維護時段執行關聯的任務。

```
Get-SSMMaintenanceWindowExecutionTaskList -WindowExecutionId
"518d5565-5969-4cca-8f0e-da3b2a638355"
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeMaintenanceWindowExecutionTasks](#)。

Get-SSMMaintenanceWindowList

以下程式碼範例顯示如何使用 Get-SSMMaintenanceWindowList。

PowerShell V4 的工具

範例 1：此範例列出了帳戶的所有維護時段。

```
Get-SSMMaintenanceWindowList
```

輸出：

```
Cutoff    : 1
Duration  : 4
Enabled   : True
Name      : My-First-Maintenance-Window
WindowId  : mw-06d59c1a07c022145
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeMaintenanceWindows](#)。

Get-SSMMaintenanceWindowTarget

以下程式碼範例顯示如何使用 Get-SSMMaintenanceWindowTarget。

PowerShell V4 的工具

範例 1：此範例列出了維護時段的所有目標。

```
Get-SSMMaintenanceWindowTarget -WindowId "mw-06cf17cbefcb4bf4f"
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeMaintenanceWindowTargets](#)。

Get-SSMMaintenanceWindowTaskList

以下程式碼範例顯示如何使用 Get-SSMMaintenanceWindowTaskList。

PowerShell V4 的工具

範例 1：此範例列出了維護時段的所有任務。

```
Get-SSMMaintenanceWindowTaskList -WindowId "mw-06cf17cbefcb4bf4f"
```

輸出：

```
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

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeMaintenanceWindowTasks](#)。

Get-SSMParameterHistory

以下程式碼範例顯示如何使用 Get-SSMParameterHistory。

PowerShell V4 的工具

範例 1：此範例列出了參數的值歷史記錄。

```
Get-SSMParameterHistory -Name "Welcome"
```

輸出：

```

Description      :
KeyId            :
LastModifiedDate : 3/3/2017 6:55:25 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type             : String
Value           : helloWorld

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetParameterHistory](#)。

Get-SSMParameterList

以下程式碼範例顯示如何使用 Get-SSMParameterList。

PowerShell V4 的工具

範例 1：此範例列出了所有參數。

```
Get-SSMParameterList
```

輸出：

```
Description      :
KeyId            :
LastModifiedDate : 3/3/2017 6:58:23 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type             : String
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeParameters](#)。

Get-SSMParameterValue

以下程式碼範例顯示如何使用 Get-SSMParameterValue。

PowerShell V4 的工具

範例 1：此範例列出了參數的多個值。

```
Get-SSMParameterValue -Name "Welcome"
```

輸出：

```
InvalidParameters Parameters
-----
{}                  {Welcome}
```

範例 2：此範例列出了值的詳細資訊。

```
(Get-SSMParameterValue -Name "Welcome").Parameters
```

輸出：

```
Name      Type      Value
```



```

-----
Welcome String Good day, Sunshine!

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [GetParameters](#)。

Get-SSMPatchBaseline

以下程式碼範例顯示如何使用 Get-SSMPatchBaseline。

PowerShell V4 的工具

範例 1：此範例列出了所有修補基準。

```
Get-SSMPatchBaseline
```

輸出：

BaselineDescription	BaselineName	BaselineId
Default Patch Baseline Provided by AWS.		arn:aws:ssm:us-west-2:123456789012:patchbaseline/pb-04fb4ae6142167966
Baseline containing all updates approved for production systems	Production-B...	pb-045f10b4f382baeda
Baseline containing all updates approved for production systems	Production-B...	pb-0a2f1059b670ebd31

範例 2：此範例列出 提供的所有修補程式基準 AWS。此範例使用的語法需要 PowerShell 版本 3 或更新版本。

```
$filter1 = @{"Key"="OWNER";Values=@("AWS")}
```

輸出：

```
Get-SSMPatchBaseline -Filter $filter1
```

範例 3：此範例列出了您作為擁有者的所有修補基準。此範例使用的語法需要 PowerShell 版本 3 或更新版本。

```
$filter1 = @{Key="OWNER";Values=@("Self")}
```

輸出：

```
Get-SSMPatchBaseline -Filter $filter1
```

範例 4：使用 PowerShell 版本 2 時，必須使用 New-Object 來建立每個標籤。

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "OWNER"
$filter1.Values = "AWS"

Get-SSMPatchBaseline -Filter $filter1
```

輸出：

BaselineDescription	BaselineName	BaselineId	DefaultBaseline
-----	-----	-----	e
Default Patch Baseline Provided by AWS.	arn:aws:ssm:us-west-2:123456789012:patchbaseline/pb-04fb4ae6142167966	AWS-DefaultPatchBaseline	True

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribePatchBaselines](#)。

Get-SSMPatchBaselineDetail

以下程式碼範例顯示如何使用 Get-SSMPatchBaselineDetail。

PowerShell V4 的工具

範例 1：此範例顯示修補基準的詳細資訊。

```
Get-SSMPatchBaselineDetail -BaselineId "pb-03da896ca3b68b639"
```

輸出：

```

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    : {}

```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetPatchBaseline](#)。

Get-SSMPatchBaselineForPatchGroup

以下程式碼範例顯示如何使用 Get-SSMPatchBaselineForPatchGroup。

PowerShell V4 的工具

範例 1：此範例顯示修補程式群組的修補基準。

```
Get-SSMPatchBaselineForPatchGroup -PatchGroup "Production"
```

輸出：

```

BaselineId          PatchGroup
-----
pb-045f10b4f382baeda Production

```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [GetPatchBaselineForPatchGroup](#)。

Get-SSMPatchGroup

以下程式碼範例顯示如何使用 Get-SSMPatchGroup。

PowerShell V4 的工具

範例 1：此範例列出了修補程式群組註冊。

```
Get-SSMPatchGroup
```

輸出：

BaselineIdentity	PatchGroup
-----	-----
Amazon.SimpleSystemsManagement.Model.PatchBaselineIdentity	Production

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribePatchGroups](#)。

Get-SSMPatchGroupState

以下程式碼範例顯示如何使用 Get-SSMPatchGroupState。

PowerShell V4 的工具

範例 1：此範例示範取得修補程式群組的高階修補程式合規摘要。

```
Get-SSMPatchGroupState -PatchGroup "Production"
```

輸出：

Instances	: 4
InstancesWithFailedPatches	: 1
InstancesWithInstalledOtherPatches	: 4
InstancesWithInstalledPatches	: 3
InstancesWithMissingPatches	: 0
InstancesWithNotApplicablePatches	: 0

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribePatchGroupState](#)。

Get-SSMResourceComplianceSummaryList

以下程式碼範例顯示如何使用 Get-SSMResourceComplianceSummaryList。

PowerShell V4 的工具

範例 1：此範例示範取得資源層級摘要計數。摘要包含有關合規和不合規狀態的資訊，以及符合 "Windows10" 之產品的詳細合規項目嚴重性計數。由於未指定該參數時，MaxResult 預設為 100，且該值無效，因此新增 MaxResult 參數，並將該值設定為 50。

```
$FilterValues = @{
    "Key"="Product"
    "Type"="EQUAL"
    "Values"="Windows10"
}

Get-SSMResourceComplianceSummaryList -Filter $FilterValues -MaxResult 50
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListResourceComplianceSummaries](#)。

Get-SSMResourceTag

以下程式碼範例顯示如何使用 Get-SSMResourceTag。

PowerShell V4 的工具

範例 1：此範例列出了維護時段的標籤。

```
Get-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
"MaintenanceWindow"
```

輸出：

```
Key    Value
---    -
Stack  Production
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ListTagsForResource](#)。

New-SSMActivation

以下程式碼範例顯示如何使用 New-SSMActivation。

PowerShell V4 的工具

範例 1：此範例示範建立受管執行個體。

```
New-SSMAutomation -DefaultInstanceName "MyWebServers" -IamRole "SSMAutomationRole" -
RegistrationLimit 10
```

輸出：

```
ActivationCode      ActivationId
-----
KWChh0xBTiwDcKE9B1KC 08e51e79-1e36-446c-8e63-9458569c1363
```

- 如需 API 詳細資訊，請參閱《Cmdlet 參考》中的 [CreateAutomation](#) (V4)。AWS Tools for PowerShell

New-SSMAssociation

以下程式碼範例顯示如何使用 New-SSMAssociation。

PowerShell V4 的工具

範例 1：此範例使用執行個體 ID 來關聯組態文件與執行個體。

```
New-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

輸出：

```
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 :
```

範例 2：此範例使用目標來關聯組態文件與執行個體。

```
$target = @{"Key"="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
```

```
New-SSMAssociation -Name "AWS-UpdateSSMAgent" -Target $target
```

輸出：

```
Name                : AWS-UpdateSSMAgent
InstanceId           :
Date                 : 3/1/2017 6:22:21 PM
Status.Name          :
Status.Date          :
Status.Message       :
Status.AdditionalInfo :
```

範例 3：此範例使用目標和參數來關聯組態文件與執行個體。

```
$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
```

輸出：

```
Name                : Configure-CloudWatch
InstanceId           :
Date                 : 5/17/2018 3:17:44 PM
Status.Name          :
Status.Date          :
Status.Message       :
Status.AdditionalInfo :
```

範例 4：此範例使用 **AWS-GatherSoftwareInventory** 建立與區域中所有執行個體的關聯。它也提供參數中要收集的自訂檔案和登錄檔位置

```
$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)"
```

輸出：

```
Name                : AWS-GatherSoftwareInventory
InstanceId          :
Date                : 6/9/2019 8:57:56 AM
Status.Name         :
Status.Date         :
Status.Message      :
Status.AdditionalInfo :
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateAssociation](#)。

New-SSMAssociationFromBatch

以下程式碼範例顯示如何使用 New-SSMAssociationFromBatch。

PowerShell V4 的工具

範例 1：此範例示範將組態文件與多個執行個體關聯。輸出會傳回成功操作和失敗操作的清單 (如適用)。

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@"AWS-UpdateSSMAgent"}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@"AWS-UpdateSSMAgent"}
New-SSMAssociationFromBatch -Entry $option1,$option2
```

輸出：

```
Failed Successful
-----
{}           {Amazon.SimpleSystemsManagement.Model.FailedCreateAssociation,
Amazon.SimpleSystemsManagement.Model.FailedCreateAsso...
```


範例 2：此範例會顯示成功操作的完整詳細資訊。

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@"AWS-UpdateSSMAgent"}}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@"AWS-UpdateSSMAgent"}}
(New-SSMAssociationFromBatch -Entry $option1,$option2).Successful
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateAssociationBatch](#)。

New-SSMDocument

以下程式碼範例顯示如何使用 New-SSMDocument。

PowerShell V4 的工具

範例 1：此範例示範在您的帳戶中建立文件。文件必須為 JSON 格式。如需有關撰寫組態文件的詳細資訊，請參閱《SSM API 參考》中的組態文件。

```
New-SSMDocument -Content (Get-Content -Raw "c:\temp\RunShellScript.json") -Name
"RunShellScript" -DocumentType "Command"
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateDocument](#)。

New-SSMMaintenanceWindow

以下程式碼範例顯示如何使用 New-SSMMaintenanceWindow。

PowerShell V4 的工具

範例 1：此範例會建立一個具有指定名稱的新維護時段，該維護時段在每個星期二下午 4 點執行 4 小時並有一個 1 小時的截止，而且允許未關聯的目標。

```
New-SSMMaintenanceWindow -Name "MyMaintenanceWindow" -Duration 4 -Cutoff 1 -
AllowUnassociatedTarget $true -Schedule "cron(0 16 ? * TUE *)"
```

輸出：

```
mw-03eb53e1ea7383998
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateMaintenanceWindow](#)。

New-SSMPatchBaseline

以下程式碼範例顯示如何使用 New-SSMPatchBaseline。

PowerShell V4 的工具

範例 1：此範例示範針對在生產環境中執行 Windows Server 2019 的受管執行個體建立修補基準，該修補基準在 Microsoft 發行修補程式七天後對修補程式進行核准。

```
$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
```

輸出：

```
pb-0z4z6221c4296b23z
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreatePatchBaseline](#)。

Register-SSMDefaultPatchBaseline

以下程式碼範例顯示如何使用 Register-SSMDefaultPatchBaseline。

PowerShell V4 的工具

範例 1：此範例示範將修補基準註冊為預設修補基準。

```
Register-SSMDefaultPatchBaseline -BaselineId "pb-03da896ca3b68b639"
```

輸出：

```
pb-03da896ca3b68b639
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [RegisterDefaultPatchBaseline](#)。

Register-SSMPatchBaselineForPatchGroup

以下程式碼範例顯示如何使用 Register-SSMPatchBaselineForPatchGroup。

PowerShell V4 的工具

範例 1：此範例示範註冊修補程式群組的修補基準。

```
Register-SSMPatchBaselineForPatchGroup -BaselineId "pb-03da896ca3b68b639" -
PatchGroup "Production"
```

輸出：

```
BaselineId          PatchGroup
-----
pb-03da896ca3b68b639 Production
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [RegisterPatchBaselineForPatchGroup](#)。

Register-SSMTargetWithMaintenanceWindow

以下程式碼範例顯示如何使用 Register-SSMTargetWithMaintenanceWindow。

PowerShell V4 的工具

範例 1：此範例示範向維護時段註冊執行個體。

```
$option1 = @{Key="InstanceIds";Values=@("i-0000293ffd8c57862")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
$option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

輸出：

```
d8e47760-23ed-46a5-9f28-927337725398
```

範例 2：此範例示範向維護時段註冊多個執行個體。

```
$option1 =
@{Key="InstanceIds";Values=@("i-0000293ffd8c57862","i-0cb2b964d3e14fd9f")}
```

```
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target $option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

輸出：

```
6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

範例 3：此範例示範使用 EC2 標籤向維護時段註冊執行個體。

```
$option1 = @{Key="tag:Environment";Values=@("Production")}  
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target $option1 -OwnerInformation "Production Web Servers" -ResourceType "INSTANCE"
```

輸出：

```
2994977e-aefb-4a71-beac-df620352f184
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [RegisterTargetWithMaintenanceWindow](#)。

Register-SSMTaskWithMaintenanceWindow

以下程式碼範例顯示如何使用 Register-SSMTaskWithMaintenanceWindow。

PowerShell V4 的工具

範例 1：此範例示範使用執行個體 ID 向維護時段註冊任務。輸出為任務 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-0000293fffd8c57862" } -TaskType "RUN_COMMAND" -  
    Priority 10 -TaskParameter $parameters
```

輸出：

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

範例 2：此範例示範使用目標 ID 向維護時段註冊任務。輸出為任務 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
```

輸出：

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

範例 3：此範例示範建立 Run Command 文件 **AWS-RunPowerShellScript** 的參數物件，並使用目標 ID 向指定維護時段建立任務。傳回輸出為任務 ID。

```
$parameters =
    [Collections.Generic.Dictionary[String,Collections.Generic.List[String]]]::new()
$parameters.Add("commands",@("ipconfig","dir env:\computername"))
$parameters.Add("executionTimeout",@(3600))

$props = @{
    WindowId = "mw-0123e4cce56ff78ae"
    ServiceRoleArn = "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
    MaxConcurrency = 1
    MaxError = 1
    TaskType = "RUN_COMMAND"
    TaskArn = "AWS-RunPowerShellScript"
    Target = @{Key="WindowTargetIds";Values="fe1234ea-56d7-890b-12f3-456b789bee0f"}
    Priority = 1
    RunCommand_Parameter = $parameters
    Name = "set-via-cmdlet"
```

```
}

Register-SSMTaskWithMaintenanceWindow @props
```

輸出：

```
f1e2ef34-5678-12e3-456a-12334c5c6cbe
```

範例 4：此範例會使用名為 `mw-123EXAMPLE456` 的文件來註冊 AWS Systems Manager Automation 任務 **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"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [RegisterTaskWithMaintenanceWindow](#)。

Remove-SSMActivation

以下程式碼範例顯示如何使用 `Remove-SSMActivation`。

PowerShell V4 的工具

範例 1：此範例示範刪除啟用。如果命令成功，則無輸出訊息。

```
Remove-SSMActivation -ActivationId "08e51e79-1e36-446c-8e63-9458569c1363"
```

- 如需 API 詳細資訊，請參閱《Cmdlet 參考》中的 [DeleteActivation](#) (V4)。AWS Tools for PowerShell

Remove-SSMAssociation

以下程式碼範例顯示如何使用 Remove-SSMAssociation。

PowerShell V4 的工具

範例 1：此範例示範刪除執行個體與文件之間的關聯。如果命令成功，則無輸出訊息。

```
Remove-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteAssociation](#)。

Remove-SSMDocument

以下程式碼範例顯示如何使用 Remove-SSMDocument。

PowerShell V4 的工具

範例 1：此範例示範刪除文件。如果命令成功，則無輸出訊息。

```
Remove-SSMDocument -Name "RunShellScript"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeleteDocument](#)。

Remove-SSMMaintenanceWindow

以下程式碼範例顯示如何使用 Remove-SSMMaintenanceWindow。

PowerShell V4 的工具

範例 1：此範例示範移除維護時段。

```
Remove-SSMMaintenanceWindow -WindowId "mw-06d59c1a07c022145"
```

輸出：

```
mw-06d59c1a07c022145
```


- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeleteMaintenanceWindow](#)。

Remove-SSMParameter

以下程式碼範例顯示如何使用 Remove-SSMParameter。

PowerShell V4 的工具

範例 1：此範例示範刪除參數。如果命令成功，則無輸出訊息。

```
Remove-SSMParameter -Name "helloWorld"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteParameter](#)。

Remove-SSMPatchBaseline

以下程式碼範例顯示如何使用 Remove-SSMPatchBaseline。

PowerShell V4 的工具

範例 1：此範例示範刪除修補基準。

```
Remove-SSMPatchBaseline -BaselineId "pb-045f10b4f382baeda"
```

輸出：

```
pb-045f10b4f382baeda
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeletePatchBaseline](#)。

Remove-SSMResourceTag

以下程式碼範例顯示如何使用 Remove-SSMResourceTag。

PowerShell V4 的工具

範例 1：此範例示範從維護時段中移除標籤。如果命令成功，則無輸出訊息。

```
Remove-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
"MaintenanceWindow" -TagKey "Production"
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [RemoveTagsFromResource](#)。

Send-SSMCommand

以下程式碼範例顯示如何使用 Send-SSMCommand。

PowerShell V4 的工具

範例 1：此範例示範在目標執行個體上執行 echo 命令。

```
Send-SSMCommand -DocumentName "AWS-RunPowerShellScript" -Parameter @{commands =
"echo helloWorld"} -Target @{Key="instanceids";Values=@"i-0cb2b964d3e14fd9f"}}
```

輸出：

```
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}
```

範例 2：此範例顯示如何執行接受巢狀參數的命令。

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [SendCommand](#)。

Start-SSMAutomationExecution

以下程式碼範例顯示如何使用 Start-SSMAutomationExecution。

PowerShell V4 的工具

範例 1：此範例示範執行指定自動化角色、AMI 來源 ID 和 Amazon EC2 執行個體角色的文件。

```
Start-SSMAutomationExecution -DocumentName AWS-UpdateLinuxAmi -
Parameter @{ 'AutomationAssumeRole'='arn:aws:iam::123456789012:role/
SSMAutomationRole'; 'SourceAmiId'='ami-f173cc91'; 'InstanceIamRole'='EC2InstanceRole' }
```

輸出：

```
3a532a4f-0382-11e7-9df7-6f11185f6dd1
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [StartAutomationExecution](#)。

Start-SSMSession

以下程式碼範例顯示如何使用 Start-SSMSession。

PowerShell V4 的工具

範例 1：此範例示範啟動與 Session Manager 工作階段目標的連線，以啟用連接埠轉送。

```
Start-SSMSession -Target 'i-064578e5e7454488f' -DocumentName 'AWS-
StartPortForwardingSession' -Parameter @{ localPortNumber = '8080'; portNumber =
'80' }
```

輸出：

```

SessionId      StreamUrl
-----
random-id0     wss://ssmmessages.amazonaws.com/v1/data-channel/random-id

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [StartSession](#)。

Stop-SSMAutomationExecution

以下程式碼範例顯示如何使用 Stop-SSMAutomationExecution。

PowerShell V4 的工具

範例 1：此範例示範停止自動化執行。如果命令成功，則無輸出訊息。

```
Stop-SSMAutomationExecution -AutomationExecutionId "4105a4fc-f944-11e6-9d32-8fb2db27a909"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [StopAutomationExecution](#)。

Stop-SSMCommand

以下程式碼範例顯示如何使用 Stop-SSMCommand。

PowerShell V4 的工具

範例 1：此範例嘗試取消命令。如果操作成功，則無輸出訊息。

```
Stop-SSMCommand -CommandId "9ded293e-e792-4440-8e3e-7b8ec5feaa38"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CancelCommand](#)。

Unregister-SSMManagedInstance

以下程式碼範例顯示如何使用 Unregister-SSMManagedInstance。

PowerShell V4 的工具

範例 1：此範例示範取消註冊受管執行個體。如果命令成功，則無輸出訊息。

```
Unregister-SSMManagedInstance -InstanceId "mi-08ab247cdf1046573"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DeregisterManagedInstance](#)。

Unregister-SSMPatchBaselineForPatchGroup

以下程式碼範例顯示如何使用 Unregister-SSMPatchBaselineForPatchGroup。

PowerShell V4 的工具

範例 1：此範例示範從修補基準中取消註冊修補程式群組。

```
Unregister-SSMPatchBaselineForPatchGroup -BaselineId "pb-045f10b4f382baeda" -
PatchGroup "Production"
```

輸出：

```
BaselineId      PatchGroup
-----
pb-045f10b4f382baeda Production
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeregisterPatchBaselineForPatchGroup](#)。

Unregister-SSMTargetFromMaintenanceWindow

以下程式碼範例顯示如何使用 Unregister-SSMTargetFromMaintenanceWindow。

PowerShell V4 的工具

範例 1：此範例示範從維護時段中移除目標。

```
Unregister-SSMTargetFromMaintenanceWindow -WindowTargetId "6ab5c208-9fc4-4697-84b7-
b02a6cc25f7d" -WindowId "mw-06cf17cbefcb4bf4f"
```

輸出：

WindowId	WindowTargetId
-----	-----
mw-06cf17cbefcb4bf4f	6ab5c208-9fc4-4697-84b7-b02a6cc25f7d

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DeregisterTargetFromMaintenanceWindow](#)。

Unregister-SSMTaskFromMaintenanceWindow

以下程式碼範例顯示如何使用 Unregister-SSMTaskFromMaintenanceWindow。

PowerShell V4 的工具

範例 1：此範例示範從維護時段中移除任務。

```
Unregister-SSMTaskFromMaintenanceWindow -WindowTaskId "f34a2c47-ddfd-4c85-a88d-72366b69af1b" -WindowId "mw-03a342e62c96d31b0"
```

輸出：

WindowId	WindowTaskId
-----	-----
mw-03a342e62c96d31b0	f34a2c47-ddfd-4c85-a88d-72366b69af1b

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeregisterTaskFromMaintenanceWindow](#)。

Update-SSMAssociation

以下程式碼範例顯示如何使用 Update-SSMAssociation。

PowerShell V4 的工具

範例 1：此範例示範以新文件版本更新關聯。

```
Update-SSMAssociation -AssociationId "93285663-92df-44cb-9f26-2292d4ecc439" -DocumentVersion "1"
```

輸出：

```

Name                : AWS-UpdateSSMAgent
InstanceId           :
Date                : 3/1/2017 6:22:21 PM
Status.Name         :
Status.Date         :
Status.Message      :
Status.AdditionalInfo :

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UpdateAssociation](#)。

Update-SSMAssociationStatus

以下程式碼範例顯示如何使用 Update-SSMAssociationStatus。

PowerShell V4 的工具

範例 1：此範例示範更新執行個體與組態文件之間的關聯狀態。

```

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"

```

輸出：

```

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

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UpdateAssociationStatus](#)。

Update-SSMDocument

以下程式碼範例顯示如何使用 Update-SSMDocument。

PowerShell V4 的工具

範例 1：此範例示範建立新版本的文件，其中包含您指定的 json 檔案的更新內容。文件必須為 JSON 格式。您可以使用 "Get-SSMDocumentVersionList" cmdlet 取得文件版本。

```
Update-SSMDocument -Name RunShellScript -DocumentVersion "1" -Content (Get-Content -Raw "c:\temp\RunShellScript.json")
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UpdateDocument](#)。

Update-SSMDocumentDefaultVersion

以下程式碼範例顯示如何使用 Update-SSMDocumentDefaultVersion。

PowerShell V4 的工具

範例 1：此範例示範更新文件的預設版本。您可以使用 "Get-SSMDocumentVersionList" cmdlet 取得可用的文件版本。

```
Update-SSMDocumentDefaultVersion -Name "RunShellScript" -DocumentVersion "2"
```

輸出：

DefaultVersion	Name
2	RunShellScript

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UpdateDocumentDefaultVersion](#)。

Update-SSMMaintenanceWindow

以下程式碼範例顯示如何使用 Update-SSMMaintenanceWindow。

PowerShell V4 的工具

範例 1：此範例示範更新維護時段的名稱。

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Name "My-Renamed-MW"
```

輸出：

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

範例 2：此範例示範啟用維護時段。

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $true
```

輸出：

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

範例 3：此範例示範停用維護時段。

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $false
```

輸出：

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : False
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UpdateMaintenanceWindow](#)。

Update-SSMManagedInstanceRole

以下程式碼範例顯示如何使用 Update-SSMManagedInstanceRole。

PowerShell V4 的工具

範例 1：此範例示範更新受管執行個體的角色。如果命令成功，則無輸出訊息。

```
Update-SSMManagedInstanceRole -InstanceId "mi-08ab247cdf1046573" -IamRole "AutomationRole"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [UpdateManagedInstanceRole](#)。

Update-SSMPatchBaseline

以下程式碼範例顯示如何使用 Update-SSMPatchBaseline。

PowerShell V4 的工具

範例 1：此範例示範將修補程式新增到現有修補基準，兩個修補程式新增為已拒絕，一個修補程式新增為已核准。

```
Update-SSMPatchBaseline -BaselineId "pb-03da896ca3b68b639" -RejectedPatch
"KB2032276","MS10-048" -ApprovedPatch "KB2124261"
```

輸出：

```
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}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [UpdatePatchBaseline](#)。

Write-SSMComplianceItem

以下程式碼範例顯示如何使用 Write-SSMComplianceItem。

PowerShell V4 的工具

範例 1：此範例為指定的受管執行個體撰寫自訂合規項目

```
$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"
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [PutComplianceItems](#)。

Write-SSMInventory

以下程式碼範例顯示如何使用 Write-SSMInventory。

PowerShell V4 的工具

範例 1：此範例示範將機架位置資訊指派給執行個體。如果命令成功，則無輸出訊息。

```
$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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [PutInventory](#)。

Write-SSMParameter

以下程式碼範例顯示如何使用 Write-SSMParameter。

PowerShell V4 的工具

範例 1：此範例示範建立參數。如果命令成功，則無輸出訊息。

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "helloWorld"
```

範例 2：此範例示範變更參數。如果命令成功，則無輸出訊息。

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "Good day, Sunshine!" -
Overwrite $true
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [PutParameter](#)。

使用 Tools for PowerShell V4 的 Amazon Translate 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 Amazon Translate 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

ConvertTo-TRNTargetLanguage

以下程式碼範例顯示如何使用 ConvertTo-TRNTargetLanguage。

PowerShell V4 的工具

範例 1：將指定的英文文字轉換為法文。要轉換的文字也可以做為 -Text 參數傳遞。

```
"Hello World" | ConvertTo-TRNTargetLanguage -SourceLanguageCode en -  
TargetLanguageCode fr
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [TranslateText](#)。

AWS WAFV2 使用 Tools for PowerShell V4 的範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 來執行動作和實作常見案例 AWS WAFV2。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執行程式碼的指示。

主題

- [動作](#)

動作

New-WAF2WebACL

以下程式碼範例顯示如何使用 New-WAF2WebACL。

PowerShell V4 的工具

範例 1：此命令會建立新的 Web ACL，名為 "waf-test"。請注意，根據服務 API 文件，「DefaultAction」是必要的屬性。因此，應該指定 '-DefaultAction-Allow' 和/或 '-DefaultAction-Block' 的值。由於 '-DefaultAction-Allow' 和 '-DefaultAction-Block' 不是必要的屬性，因此值 '@{}' 可以用作預留位置，如上述範例所示。

```
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 @{}
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [CreateWebAcl](#)。

使用 Tools for PowerShell V4 的 WorkSpaces 範例

下列程式碼範例示範如何使用 AWS Tools for PowerShell V4 搭配 WorkSpaces 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例都包含完整原始程式碼的連結，您可以在其中找到如何在內容中設定和執程式碼的指示。

主題

- [動作](#)

動作

Approve-WKSIpRule

以下程式碼範例顯示如何使用 Approve-WKSIpRule。

PowerShell V4 的工具

範例 1：此範例會將規則新增至現有的 IP 群組

```
$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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [AuthorizeIpRules](#)。

Copy-WKSWorkspaceImage

以下程式碼範例顯示如何使用 Copy-WKSWorkspaceImage。

PowerShell V4 的工具

範例 1：此範例會將具有 us-west-2 指定 ID 的工作區映像複製到名為 "CopiedImageTest" 的目前區域

```
Copy-WKSWorkspaceImage -Name CopiedImageTest -SourceRegion us-west-2 -SourceImageId
wsi-djfoedhw6
```

輸出：

```
wsi-456abaqfe
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CopyWorkspaceImage](#)。

Edit-WKSClientProperty

以下程式碼範例顯示如何使用 Edit-WKSClientProperty。

PowerShell V4 的工具

範例 1：此範例會啟用工作區用戶端的重新連線

```
Edit-WKSClientProperty -Region us-west-2 -ClientProperties_ReconnectEnabled  
"ENABLED" -ResourceId d-123414a369
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ModifyClientProperties](#)。

Edit-WKSSelfServicePermission

以下程式碼範例顯示如何使用 Edit-WKSSelfServicePermission。

PowerShell V4 的工具

範例 1：此範例可讓自助服務許可變更指定目錄的運算類型和增加磁碟區大小

```
Edit-WKSSelfservicePermission -Region us-west-2 -ResourceId  
d-123454a369 -SelfservicePermissions_ChangeComputeType ENABLED -  
SelfservicePermissions_IncreaseVolumeSize ENABLED
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [ModifySelfservicePermissions](#)。

Edit-WKSWorkspaceAccessProperty

以下程式碼範例顯示如何使用 Edit-WKSWorkspaceAccessProperty。

PowerShell V4 的工具

範例 1：此範例可為指定的目錄啟用 Android 和 Chrome 作業系統上的工作區存取


```
Edit-WKSWorkspaceAccessProperty -Region us-west-2 -ResourceId  
d-123454a369 -WorkspaceAccessProperties_DeviceTypeAndroid ALLOW -  
WorkspaceAccessProperties_DeviceTypeChromeOs ALLOW
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [ModifyWorkspaceAccessProperties](#)。

Edit-WKSWorkspaceCreationProperty

以下程式碼範例顯示如何使用 Edit-WKSWorkspaceCreationProperty。

PowerShell V4 的工具

範例 1：此範例可讓網際網路存取和維護模式在建立工作區時成為預設值

```
Edit-WKSWorkspaceCreationProperty -Region us-west-2 -ResourceId  
d-123454a369 -WorkspaceCreationProperties_EnableInternetAccess $true -  
WorkspaceCreationProperties_EnableMaintenanceMode $true
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ModifyWorkspaceCreationProperties](#)。

Edit-WKSWorkspaceProperty

以下程式碼範例顯示如何使用 Edit-WKSWorkspaceProperty。

PowerShell V4 的工具

範例 1：此範例會將指定工作區的工作區執行模式屬性變更為自動停止

```
Edit-WKSWorkspaceProperty -WorkspaceId ws-w361s100v -Region us-west-2 -  
WorkspaceProperties_RunningMode AUTO_STOP
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ModifyWorkspaceProperties](#)。

Edit-WKSWorkspaceState

以下程式碼範例顯示如何使用 Edit-WKSWorkspaceState。

PowerShell V4 的工具

範例 1：此範例會將指定工作區的狀態變更為可用

```
Edit-WKSSpaceState -WorkspaceId ws-w361s100v -Region us-west-2 -WorkspaceState
AVAILABLE
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [ModifyWorkspaceState](#)。

Get-WKSClientProperty

以下程式碼範例顯示如何使用 Get-WKSClientProperty。

PowerShell V4 的工具

範例 1：此範例取得指定目錄之工作區用戶端的用戶端屬性

```
Get-WKSClientProperty -ResourceId d-223562a123
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeClientProperties](#)。

Get-WKSIpGroup

以下程式碼範例顯示如何使用 Get-WKSIpGroup。

PowerShell V4 的工具

範例 1：此範例會取得指定區域中指定 IP 群組的詳細資訊

```
Get-WKSIpGroup -Region us-east-1 -GroupId wsipg-8m1234v45
```

輸出：

```
GroupDesc GroupId      GroupName UserRules
-----
wsipg-8m1234v45 TestGroup {Amazon.WorkSpaces.Model.IpRuleItem,
Amazon.WorkSpaces.Model.IpRuleItem}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeGroups](#)。

Get-WKSTag

以下程式碼範例顯示如何使用 Get-WKSTag。

PowerShell V4 的工具

範例 1：此範例會擷取指定工作區的標籤

```
Get-WKSTag -WorkspaceId ws-w361s234r -Region us-west-2
```

輸出：

Key	Value
---	-----
auto-delete	no
purpose	Workbench

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeTags](#)。

Get-WKSWorkspace

以下程式碼範例顯示如何使用 Get-WKSWorkspace。

PowerShell V4 的工具

範例 1：擷取管道中所有 WorkSpaces 的詳細資訊。

```
Get-WKSWorkspace
```

輸出：

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

```

範例 2：此命令顯示 **us-west-2** 區域中 **WorkspaceProperties** 工作區的字屬性值。如需子屬性的詳細資訊 **WorkspaceProperties**，請參閱 https://docs.aws.amazon.com/workspaces/latest/api/API_WorkspaceProperties.html。

```
(Get-WKSWorkspace -Region us-west-2 -WorkspaceId ws-xdaf7hc9s).WorkspaceProperties
```

輸出：

```

ComputeTypeName           : STANDARD
RootVolumeSizeGib         : 80
RunningMode                : AUTO_STOP
RunningModeAutoStopTimeoutInMinutes : 60
UserVolumeSizeGib         : 50

```

範例 3：此命令顯示 **us-west-2** 區域中 **RootVolumeSizeGibWorkspaceProperties** 工作區的字屬性值。GiB 的根磁碟區大小為 80。

```
(Get-WKSWorkspace -Region us-west-2 -WorkspaceId ws-xdaf7hc9s).WorkspaceProperties.RootVolumeSizeGib
```

輸出：

```
80
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeWorkspaces](#)。

Get-WKSWorkspaceBundle

以下程式碼範例顯示如何使用 `Get-WKSWorkspaceBundle`。

PowerShell V4 的工具

範例 1：此範例會擷取目前區域中所有工作區套件的詳細資訊

```
Get-WKSWorkspaceBundle
```

輸出：

```
BundleId       : wsb-sfhdgv342
ComputeType    : Amazon.WorkSpaces.Model.ComputeType
Description     : This bundle is custom
ImageId        : wsi-235aeqges
LastUpdatedTime : 12/26/2019 06:44:07
Name           : CustomBundleTest
Owner          : 233816212345
RootStorage    : Amazon.WorkSpaces.Model.RootStorage
UserStorage    : Amazon.WorkSpaces.Model.UserStorage
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeWorkspaceBundles](#)。

Get-WKSWorkspaceDirectory

以下程式碼範例顯示如何使用 Get-WKSWorkspaceDirectory。

PowerShell V4 的工具

範例 1：此範例列出已註冊目錄的目錄詳細資訊

```
Get-WKSWorkspaceDirectory
```

輸出：

```
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

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeWorkspaceDirectories](#)。

Get-WKSWorkspaceImage

以下程式碼範例顯示如何使用 Get-WKSWorkspaceImage。

PowerShell V4 的工具

範例 1：此範例會擷取區域中所有影像的所有詳細資訊

```
Get-WKSWorkspaceImage
```

輸出：

```

Description      : This image is copied from another image
ErrorCode        :
ErrorMessage     :
ImageId          : wsi-345ahdjgo
Name             : CopiedImageTest
OperatingSystem  : Amazon.WorkSpaces.Model.OperatingSystem
RequiredTenancy  : DEFAULT
State           : AVAILABLE

```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DescribeWorkspaceImages](#)。

Get-WKSWorkspaceSnapshot

以下程式碼範例顯示如何使用 Get-WKSWorkspaceSnapshot。

PowerShell V4 的工具

範例 1：此範例顯示為指定工作區建立的最新快照時間戳記

```
Get-WKSWorkspaceSnapshot -WorkspaceId ws-w361s100v
```

輸出：

RebuildSnapshots	RestoreSnapshots
-----	-----
{Amazon.WorkSpaces.Model.Snapshot}	{Amazon.WorkSpaces.Model.Snapshot}

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [DescribeWorkspaceSnapshots](#)。

Get-WKSWorkspacesConnectionStatus

以下程式碼範例顯示如何使用 Get-WKSWorkspacesConnectionStatus。

PowerShell V4 的工具

範例 1：此範例會擷取指定工作區的連線狀態

```
Get-WKSWorkspacesConnectionStatus -WorkspaceId ws-w123s234r
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DescribeWorkspacesConnectionStatus](#)。

New-WKSIpGroup

以下程式碼範例顯示如何使用 New-WKSIpGroup。

PowerShell V4 的工具

範例 1：此範例會建立名為 FreshEmptyIpGroup 的空 Ip 群組

```
New-WKSIpGroup -GroupName "FreshNewIPGroup"
```

輸出：

```
wsipg-w45rty4ty
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateIpGroup](#)。

New-WKSTag

以下程式碼範例顯示如何使用 New-WKSTag。

PowerShell V4 的工具

範例 1：此範例會將新標籤新增至名為 **ws-wsname** 的工作區。標籤的索引鍵為 "Name"，索引鍵值為 **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
```

範例 2：此範例會將多個標籤新增至名為 **ws-wsname** 的工作區。一個標籤的索引鍵為「Name」且索引鍵值為 **AWS_Workspace**；另一個標籤的標籤索引鍵為「Stage」且索引鍵值為「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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [CreateTags](#)。

New-WKSWorkspace

以下程式碼範例顯示如何使用 New-WKSWorkspace。

PowerShell V4 的工具

範例 1：為提供的套件、目錄和使用者建立 Workspace。

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId" =  
"d-1a2b3c4d"; "UserName" = "USERNAME"}
```

範例 2：此範例會建立多個 WorkSpaces

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId"  
= "d-1a2b3c4d"; "UserName" = "USERNAME_1"},@{"BundleID" = "wsb-1a2b3c4d";  
"DirectoryId" = "d-1a2b3c4d"; "UserName" = "USERNAME_2"}
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [CreateWorkspaces](#)。

Register-WKSipGroup

以下程式碼範例顯示如何使用 Register-WKSipGroup。

PowerShell V4 的工具

範例 1：此範例會向指定的目錄註冊指定的 IP 群組

```
Register-WKSipGroup -GroupId wsipg-23ahsdres -DirectoryId d-123412e123
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [AssociateIpGroups](#)。

Register-WKSWorkspaceDirectory

以下程式碼範例顯示如何使用 Register-WKSWorkspaceDirectory。

PowerShell V4 的工具

範例 1：此範例會註冊 Workspaces Service 的指定目錄

```
Register-WKSWorkspaceDirectory -DirectoryId d-123412a123 -EnableWorkDoc $false
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [RegisterWorkspaceDirectory](#)。

Remove-WKSIpGroup

以下程式碼範例顯示如何使用 Remove-WKSIpGroup。

PowerShell V4 的工具

範例 1：此範例會刪除指定的 IP 群組

```
Remove-WKSIpGroup -GroupId wsipg-32fhgtred
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeletelpGroup](#)。

Remove-WKSTag

以下程式碼範例顯示如何使用 Remove-WKSTag。

PowerShell V4 的工具

範例 1：此範例會移除與工作區相關聯的標籤

```
Remove-WKSTag -ResourceId ws-w10b3abcd -TagKey "Type"
```

輸出：

```
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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteTags](#)。

Remove-WKSWorkspace

以下程式碼範例顯示如何使用 Remove-WKSWorkspace。

PowerShell V4 的工具

範例 1：終止多個 WorkSpaces。使用 -Force 切換會停止 cmdlet 提示確認。

```
Remove-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0" -Force
```

範例 2：擷取所有 WorkSpaces 的集合，並將 IDs 輸送至 Remove-WKSWorkspace 的 -WorkspaceId 參數，以終止所有 WorkSpaces。Cmdlet 會在每個 Workspace 終止之前提示。若要隱藏確認提示，請新增 -Force 切換。

```
Get-WKSWorkspaces | Remove-WKSWorkspace
```

範例 3：此範例示範如何傳遞定義要終止之 WorkSpaces 的 TerminateRequest 物件。除非也指定 -Force 切換參數，否則 cmdlet 會在繼續之前提示確認。

```
$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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [TerminateWorkspaces](#)。

Reset-WKSWorkspace

以下程式碼範例顯示如何使用 Reset-WKSWorkspace。

PowerShell V4 的工具

範例 1：重建指定的 Workspace。

```
Reset-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

範例 2：擷取所有 WorkSpaces 的集合，並將 IDs 輸送至 Reset-WKSWorkspace 的 -WorkspaceId 參數，導致 WorkSpaces 重建。

```
Get-WKSWorkspaces | Reset-WKSWorkspace
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [RebuildWorkspaces](#)。

Restart-WKSWorkspace

以下程式碼範例顯示如何使用 Restart-WKSWorkspace。

PowerShell V4 的工具

範例 1：重新啟動指定的 Workspace。

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

範例 2：重新啟動多個 WorkSpaces。

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d","ws-5a6b7c8d"
```

範例 3：擷取所有 WorkSpaces 的集合，並將 IDs 輸送至 Restart-WKSWorkspace 的 -WorkspaceId 參數，導致 WorkSpaces 重新啟動。

```
Get-WKSWorkspaces | Restart-WKSWorkspace
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [RebootWorkspaces](#)。

Stop-WKSWorkspace

以下程式碼範例顯示如何使用 Stop-WKSWorkspace。

PowerShell V4 的工具

範例 1：停止多個 WorkSpaces。

```
Stop-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0"
```

範例 2：擷取所有 WorkSpaces 的集合，並將 IDs 輸送至 Stop-WKSWorkspace 的 -WorkspaceId 參數，導致 WorkSpaces 停止。

```
Get-WKSWorkspaces | Stop-WKSWorkspace
```

範例 3：此範例示範如何傳遞 StopRequest 物件，定義要停止的 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
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet 參考 (V4) 中的 [StopWorkspaces](#)。

Unregister-WKSIpGroup

以下程式碼範例顯示如何使用 Unregister-WKSIpGroup。

PowerShell V4 的工具

範例 1：此範例會從指定的目錄取消註冊指定的 IP 群組

```
Unregister-WKSIpGroup -GroupId wsipg-12abcdphq -DirectoryId d-123454b123
```

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell 《Cmdlet Reference (V4)》中的 [DisassociateIpGroups](#)。

此 AWS 產品或服務的安全性

雲端安全是 Amazon Web Services (AWS) 最重視的一環。身為 AWS 客戶的您，將能從資料中心和網路架構的建置中獲益，以滿足組織最為敏感的安全要求。安全性是 AWS 與您之間共同責任。[共同責任模型](#) 將此描述為雲端本身的安全和雲端內部的安全。

雲端的安全性 – AWS 負責保護執行 AWS 雲端中提供的所有服務的基礎設施，並為您提供可安全使用的服務。我們的安全責任是最高優先順序 AWS，我們的安全有效性由第三方稽核人員定期測試和驗證，作為[AWS 合規計劃](#)的一部分。

雲端的安全性 – 您的責任取決於您使用 AWS 的服務，以及其他因素，包括資料的敏感度、組織的需求，以及適用的法律和法規。

此 AWS 產品或服務會透過其支援的特定 Amazon Web Services (AWS) 服務，遵循[共同責任模型](#)。如需 AWS 服務安全資訊，請參閱[AWS 服務安全文件頁面](#)，以及[AWS 合規計劃在 AWS 合規工作範圍內的服務](#)。

主題

- [此 AWS 產品或服務中的資料保護](#)
- [身分和存取權管理](#)
- [此 AWS 產品或服務的合規驗證](#)
- [在 Tools for PowerShell 中強制執行最低 TLS 版本](#)
- [適用於 PowerShell 的工具的其他安全考量事項](#)

此 AWS 產品或服務中的資料保護

AWS [共同責任模型](#)適用於此 AWS 產品或服務中的資料保護。如此模型所述，AWS 負責保護執行所有的全域基礎設施 AWS 雲端。您負責維護在此基礎設施上託管內容的控制權。您也同時負責所使用 AWS 服務的安全組態和管理任務。如需資料隱私權的詳細資訊，請參閱[資料隱私權常見問答集](#)。如需有關歐洲資料保護的相關資訊，請參閱 AWS 安全性部落格上的 [AWS 共同的責任模型和 GDPR](#) 部落格文章。

基於資料保護目的，建議您保護 AWS 帳戶登入資料，並使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 設定個別使用者。如此一來，每個使用者都只會獲得授與完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。

- 使用 SSL/TLS 與 AWS 資源通訊。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 使用 設定 API 和使用者活動記錄 AWS CloudTrail。如需有關使用 CloudTrail 追蹤擷取 AWS 活動的資訊，請參閱AWS CloudTrail 《使用者指南》中的[使用 CloudTrail 追蹤](#)。
- 使用 AWS 加密解決方案，以及其中的所有預設安全控制 AWS 服務。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Amazon S3 的敏感資料。
- 如果您在 AWS 透過命令列界面或 API 存取 時需要 FIPS 140-3 驗證的密碼編譯模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的更多相關資訊，請參閱[聯邦資訊處理標準 \(FIPS\) 140-3](#)。

我們強烈建議您絕對不要將客戶的電子郵件地址等機密或敏感資訊，放在標籤或自由格式的文字欄位中，例如名稱欄位。這包括當您使用此 AWS 產品或服務，或使用主控台、API AWS CLI或 AWS SDKs的其他 AWS 服務 時。您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。如果您提供外部伺服器的 URL，我們強烈建議請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

資料加密

任何安全服務都有一項重要功能，就是當資訊處於非使用中狀態時，就會將資訊加密。

靜態加密

除了代表 AWS 使用者與服務互動所需的登入資料之外，本身 AWS Tools for PowerShell 不會存放任何客戶資料。

如果您使用 AWS Tools for PowerShell 叫用將客戶資料傳輸至本機電腦以供儲存 AWS 的服務，請參閱該服務使用者指南中的安全與合規章節，以取得如何存放、保護和加密該資料的資訊。

傳輸中加密

根據預設，從執行 AWS Tools for PowerShell AWS 和服務端點的用戶端電腦傳輸的所有資料都會透過 HTTPS/TLS 連線傳送所有內容來加密。

您不須採取任何行動即可啟用 HTTPS/TLS。它隨時處於啟用的狀態。

身分和存取權管理

AWS Identity and Access Management (IAM) 是 AWS 服務，可協助管理員安全地控制對 AWS 資源的存取。IAM 管理員可控制誰可以進行驗證（登入）和授權（具有許可）來使用 AWS 資源。IAM 是 AWS 服務 您可以免費使用的。

主題

- [目標對象](#)
- [使用身分驗證](#)
- [使用政策管理存取權](#)
- [AWS 服務 如何使用 IAM](#)
- [對 AWS 身分和存取進行故障診斷](#)

目標對象

使用 AWS Identity and Access Management (IAM) 的方式會有所不同，取決於您在 中執行的工作 AWS。

服務使用者 – 如果您使用 AWS 服務 執行任務，管理員會為您提供所需的登入資料和許可。當您使用更多 AWS 功能來執行工作時，您可能需要額外的許可。了解存取許可的管理方式可協助您向管理員請求正確的許可。如果您無法存取 中的功能 AWS，請參閱 [對 AWS 身分和存取進行故障診斷](#)或 AWS 服務 您正在使用的 使用者指南。

服務管理員 – 如果您負責公司 AWS 的資源，您可能擁有 的完整存取權 AWS。您的任務是判斷服務使用者應該存取哪些 AWS 功能和資源。接著，您必須將請求提交給您的 IAM 管理員，來變更您服務使用者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要進一步了解貴公司如何使用 IAM AWS，請參閱您正在使用的 使用者指南 AWS 服務。

IAM 管理員：如果您是 IAM 管理員，建議您掌握如何撰寫政策以管理 AWS存取權的詳細資訊。若要檢視您可以在 IAM 中使用的以 AWS 身分為基礎的政策範例，請參閱 AWS 服務 您正在使用的 使用者指南。

使用身分驗證

身分驗證是您 AWS 使用身分憑證登入 的方式。您必須以 AWS 帳戶根使用者身分、IAM 使用者身分或擔任 IAM 角色來驗證 (登入 AWS)。

您可以使用透過身分來源提供的憑證，以聯合身分 AWS 身分身分登入。AWS IAM Identity Center (IAM Identity Center) 使用者、您公司的單一登入身分驗證，以及您的 Google 或 Facebook 登入資料，都是聯合身分的範例。您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。當您使用聯合 AWS 身分存取 時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需登入的詳細資訊 AWS，請參閱AWS 登入 《使用者指南》中的[如何登入您的 AWS 帳戶](#)。

如果您以 AWS 程式設計方式存取，AWS 會提供軟體開發套件 (SDK) 和命令列界面 (CLI)，以使用您的憑證以密碼編譯方式簽署您的請求。如果您不使用 AWS 工具，則必須自行簽署請求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱《IAM 使用者指南》中的[適用於 API 請求的 AWS Signature 第 4 版](#)。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如，AWS 建議您使用多重驗證 (MFA) 來提高帳戶的安全性。如需更多資訊，請參閱《AWS IAM Identity Center 使用者指南》中的[多重要素驗證](#)和《IAM 使用者指南》中的[IAM 中的 AWS 多重要素驗證](#)。

AWS 帳戶 根使用者

建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取帳戶中的所有 AWS 服務和資源。此身分稱為 AWS 帳戶 Theroot 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱 IAM 使用者指南中的[需要根使用者憑證的任務](#)。

聯合身分

最佳實務是，要求人類使用者，包括需要管理員存取權的使用者，使用聯合身分提供者 AWS 服務來使用臨時憑證來存取。

聯合身分是您企業使用者目錄、Web 身分提供者、AWS Directory Service、Identity Center 目錄或任何使用透過身分來源提供的登入資料 AWS 服務存取的使用者。當聯合身分存取時 AWS 帳戶，它們會擔任角色，而角色會提供臨時登入資料。

對於集中式存取權管理，我們建議您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中建立使用者和群組，也可以連接並同步到您自己的身分來源中的一組使用者 AWS 帳戶和群組，以便在所有和應用程式中使用。如需 IAM Identity Center 的詳細資訊，請參閱 AWS IAM Identity Center 使用者指南中的[什麼是 IAM Identity Center ?](#)。

IAM 使用者和群組

[IAM 使用者](#)是中的身分 AWS 帳戶，具有單一人員或應用程式的特定許可。建議您盡可能依賴臨時憑證，而不是擁有建立長期憑證 (例如密碼和存取金鑰) 的 IAM 使用者。但是如果特定使用案例需要擁有長期憑證的 IAM 使用者，建議您輪換存取金鑰。如需更多資訊，請參閱[IAM 使用者指南](#)中的為需要長期憑證的使用案例定期輪換存取金鑰。

[IAM 群組](#)是一種指定 IAM 使用者集合的身分。您無法以群組身分簽署。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的程序變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供臨時憑證。如需更多資訊，請參閱《IAM 使用者指南》中的 [IAM 使用者的使用案例](#)。

IAM 角色

[IAM 角色](#)是 中具有特定許可 AWS 帳戶 的身分。它類似 IAM 使用者，但不與特定的人員相關聯。若要暫時在 中擔任 IAM 角色 AWS Management Console，您可以從[使用者切換至 IAM 角色（主控台）](#)。您可以透過呼叫 AWS CLI 或 AWS API 操作或使用自訂 URL 來擔任角色。如需使用角色的方法詳細資訊，請參閱《IAM 使用者指南》中的[擔任角色的方法](#)。

使用臨時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 — 如需向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需有關聯合角色的相關資訊，請參閱《[IAM 使用者指南](#)》中的為第三方身分提供者 (聯合) 建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的[許可集](#)。
- 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。
- 跨帳戶存取權：您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的主體) 存取您帳戶的資源。角色是授予跨帳戶存取權的主要方式。不過，對於某些 AWS 服務，您可以直接將政策連接到資源 (而不是使用角色做為代理)。如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 [IAM 中的跨帳戶資源存取](#)。
- 跨服務存取 – 有些 AWS 服務 使用其他 中的功能 AWS 服務。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件儲存在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
- 轉送存取工作階段 (FAS) – 當您使用 IAM 使用者或角色在其中執行動作時 AWS，您會被視為委託人。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 使用呼叫的委託人許可 AWS 服務，結合 AWS 服務 請求向下游服務提出請求。只有當服務收到需要與其他 AWS 服務 或 資源互動才能完成的請求時，才會提出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱 [《轉發存取工作階段》](#)。
- 服務角色 – 服務角色是服務擔任的 [IAM 角色](#)，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱《IAM 使用者指南》中的[建立角色以委派許可權給 AWS 服務](#)。

- 服務連結角色 – 服務連結角色是連結至的一種服務角色 AWS 服務。服務可以擔任代表您執行動作的角色。服務連結角色會出現在您的 中 AWS 帳戶，並由服務擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。
- 在 Amazon EC2 上執行的應用程式 – 您可以使用 IAM 角色來管理在 EC2 執行個體上執行之應用程式的臨時登入資料，以及提出 AWS CLI 或 AWS API 請求。這是在 EC2 執行個體內儲存存取金鑰的較好方式。若要將 AWS 角色指派給 EC2 執行個體並將其提供給其所有應用程式，您可以建立連接至執行個體的執行個體描述檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得臨時憑證。如需詳細資訊，請參閱《IAM 使用者指南》中的[使用 IAM 角色來授予許可權給 Amazon EC2 執行個體上執行的應用程式](#)。

使用政策管理存取權

您可以透過建立政策並將其連接到身分或資源 AWS 來控制 AWS 中的存取。政策是 中的物件，當與身分或資源相關聯時，AWS 會定義其許可。當委託人（使用者、根使用者或角色工作階段）發出請求時，會 AWS 評估這些政策。政策中的許可決定是否允許或拒絕請求。大多數政策會以 JSON 文件 AWS 形式存放在 中。如需 JSON 政策文件結構和內容的詳細資訊，請參閱 IAM 使用者指南中的[JSON 政策概觀](#)。

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 iam:GetRole 動作的政策。具有該政策的使用者可以從 AWS Management Console AWS CLI、或 API AWS 取得角色資訊。

身分型政策

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的[透過客戶管理政策定義自訂 IAM 許可](#)。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管政策是獨立的政策，您可以連接到 中的多個使用者、群組和角色 AWS 帳戶。受管政策包括 AWS 受管政策和客戶受管政策。如需了解如何在受管政策及內嵌政策之間選擇，請參閱《IAM 使用者指南》中的[在受管政策和內嵌政策間選擇](#)。

資源型政策

資源型政策是連接到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中[指定主體](#)。委託人可以包含帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

資源型政策是位於該服務中的內嵌政策。您無法在資源型政策中使用來自 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

Amazon S3 AWS WAF 和 Amazon VPC 是支援 ACLs 的服務範例。如需進一步了解 ACL，請參閱 Amazon Simple Storage Service 開發人員指南中的[存取控制清單 \(ACL\) 概觀](#)。

其他政策類型

AWS 支援其他較不常見的政策類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 許可界限 – 許可範圍是一種進階功能，可供您設定身分型政策能授予 IAM 實體 (IAM 使用者或角色) 的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 Principal 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可界限的詳細資訊，請參閱 IAM 使用者指南中的[IAM 實體許可界限](#)。
- 服務控制政策 (SCPs) – SCPs 是 JSON 政策，可指定中組織或組織單位 (OU) 的最大許可 AWS Organizations。AWS Organizations 是一種用於分組和集中管理您企業擁有 AWS 帳戶之多個的服務。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 會限制成員帳戶中實體的許可，包括每個實體 AWS 帳戶根使用者。如需 Organizations 和 SCP 的詳細資訊，請參閱《AWS Organizations 使用者指南》中的[服務控制政策](#)。
- 資源控制政策 (RCP) - RCP 是 JSON 政策，可用來設定您帳戶中資源的可用許可上限，採取這種方式就不需要更新附加至您所擁有的每個資源的 IAM 政策。RCP 會限制成員帳戶中資源的許可，並可能影響身分的有效許可，包括 AWS 帳戶根使用者，無論它們是否屬於您的組織。如需 Organizations 和 RCPs 的詳細資訊，包括支援 RCPs AWS 服務的清單，請參閱 AWS Organizations 《使用者指南》中的[資源控制政策 \(RCPs\)](#)。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過撰寫程式的方式建立角色或聯合使用者的暫時工作階段時，做為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作

階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需詳細資訊，請參閱 IAM 使用者指南中的[工作階段政策](#)。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。若要了解如何 AWS 在涉及多個政策類型時決定是否允許請求，請參閱《IAM 使用者指南》中的[政策評估邏輯](#)。

AWS 服務 如何使用 IAM

若要深入了解 如何使用 AWS 服務 大多數 IAM 功能，請參閱《IAM 使用者指南》中的與 IAM [AWS 搭配使用的 服務](#)。

若要了解如何 AWS 服務 搭配 IAM 使用特定，請參閱相關服務使用者指南的安全章節。

對 AWS 身分和存取進行故障診斷

使用以下資訊來協助您診斷和修正使用 AWS 和 IAM 時可能遇到的常見問題。

主題

- [我無權在 中執行動作 AWS](#)
- [我未獲得執行 iam:PassRole 的授權](#)
- [我想要允許 以外的人員 AWS 帳戶 存取我的 AWS 資源](#)

我無權在 中執行動作 AWS

如果您收到錯誤，告知您未獲授權執行動作，您的政策必須更新，允許您執行動作。

下列範例錯誤會在mateojackson IAM 使用者嘗試使用主控台檢視一個虛構 *my-example-widget* 資源的詳細資訊，但卻無虛構 *awes:GetWidget* 許可時發生。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
awes:GetWidget on resource: my-example-widget
```

在此情況下，必須更新 mateojackson 使用者的政策，允許使用 *awes:GetWidget* 動作存取 *my-example-widget* 資源。

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

我未獲得執行 iam:PassRole 的授權

如果您收到錯誤，告知您未獲授權執行 iam:PassRole 動作，您的政策必須更新，允許您將角色傳遞給 AWS。

有些 AWS 服務可讓您將現有角色傳遞給該服務，而不是建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

名為 marymajor 的 IAM 使用者嘗試使用主控台在 AWS 中執行動作時，發生下列範例錯誤。但是，動作要求服務具備服務角色授予的許可。Mary 沒有將角色傳遞至該服務的許可。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

在這種情況下，Mary 的政策必須更新，允許她執行 iam:PassRole 動作。

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

我想要允許以外的人員 AWS 帳戶 存取我的 AWS 資源

您可以建立一個角色，讓其他帳戶中的使用者或您組織外部的人員存取您的資源。您可以指定要允許哪些信任物件取得該角色。針對支援基於資源的政策或存取控制清單 (ACL) 的服務，您可以使用那些政策來授予人員存取您的資源的許可。

如需進一步了解，請參閱以下內容：

- 若要了解 是否 AWS 支援這些功能，請參閱 [AWS 服務 如何使用 IAM](#)。
- 若要了解如何在您擁有 AWS 帳戶 的 資源之間提供存取權，請參閱《[IAM 使用者指南](#)》中的在您擁有 AWS 帳戶 的另一個 中為 IAM 使用者提供存取權。
- 若要了解如何將資源的存取權提供給第三方 AWS 帳戶，請參閱《IAM 使用者指南》中的[將存取權提供給第三方 AWS 帳戶 擁有](#)。
- 如需了解如何透過聯合身分提供存取權，請參閱 IAM 使用者指南中的[將存取權提供給在外部進行身分驗證的使用者 \(聯合身分\)](#)。
- 如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 [IAM 中的跨帳戶資源存取](#)。

此 AWS 產品或服務的合規驗證

若要了解 是否 AWS 服務 在特定合規計劃的範圍內，請參閱[AWS 服務 合規計劃範圍內](#)然後選擇您感興趣的合規計劃。如需一般資訊，請參閱 [AWS Compliance Programs](#)。

您可以使用 下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱[下載報告 in AWS Artifact](#)

您使用 時的合規責任 AWS 服務 取決於資料的機密性、您公司的合規目標，以及適用的法律和法規。AWS 提供下列資源來協助合規：

- [安全合規與治理](#) - 這些解決方案實作指南內容討論了架構考量，並提供部署安全與合規功能的步驟。
- [HIPAA 合格服務參考](#) - 列出 HIPAA 合格服務。並非所有 AWS 服務 都符合 HIPAA 資格。
- [AWS 合規資源](#) - 此工作手冊和指南集合可能適用於您的產業和位置。
- [AWS 客戶合規指南](#) - 透過合規的角度了解共同責任模型。本指南摘要說明跨多個架構（包括國家標準技術研究所 (NIST)、支付卡產業安全標準委員會 (PCI) 和國際標準化組織 (ISO)) 保護 AWS 服務 和映射指南至安全控制的最佳實務。
- 《AWS Config 開發人員指南》中的[使用規則評估資源](#) - AWS Config 服務會評估資源組態符合內部實務、產業準則和法規的程度。
- [AWS Security Hub](#) - 這 AWS 服務 可讓您全面檢視其中的安全狀態 AWS。Security Hub 使用安全控制，可評估您的 AWS 資源並檢查您的法規遵循是否符合安全業界標準和最佳實務。如需支援的服務和控制清單，請參閱「[Security Hub 控制參考](#)」。
- [Amazon GuardDuty](#) - 這會監控您的環境是否有可疑和惡意活動，以 AWS 服務 偵測對您 AWS 帳戶、工作負載、容器和資料的潛在威脅。GuardDuty 可滿足特定合規架構所規定的入侵偵測需求，以協助您因應 PCI DSS 等各種不同的合規需求。
- [AWS Audit Manager](#) - 這 AWS 服務 可協助您持續稽核 AWS 用量，以簡化您管理風險和符合法規和業界標準的方式。

此 AWS 產品或服務會透過其支援的特定 Amazon Web Services (AWS) 服務，遵循[共同責任模型](#)。如需 AWS 服務安全資訊，請參閱[AWS 服務安全文件頁面](#)，以及[AWS 合規計劃在 AWS 合規工作範圍內的服務](#)。

在 Tools for PowerShell 中強制執行最低 TLS 版本

若要在與服務通訊時提高安全性 AWS，您應該將 Tools for PowerShell 設定為使用適當的 TLS 版本。有關如何執行此操作的詳細資訊，請參閱[適用於 .NET 的 AWS SDK 開發人員指南](#)中的[強制執行最低 TLS 版本](#)。

適用於 PowerShell 的工具的其他安全考量事項

除了先前章節涵蓋的安全主題之外，本主題還包含安全考量事項。

記錄敏感資訊

此工具的某些操作可能會傳回可能被視為敏感的資訊，包括來自環境變數的資訊。在某些情況下，此資訊的暴露可能代表安全風險；例如，資訊可能包含在持續整合和持續部署 (CI/CD) 日誌中。因此，當您在日誌中包含此類輸出時，請務必檢閱，並在不需要時隱藏輸出。如需保護敏感資料的其他資訊，請參閱[此 AWS 產品或服務中的資料保護](#)。

請考慮下列最佳實務：

- 請勿使用環境變數來存放無伺服器資源的敏感值。反之，請讓無伺服器程式碼以程式設計方式從秘密存放區擷取秘密（例如）AWS Secrets Manager。
- 檢閱建置日誌的內容，以確保其中不包含敏感資訊。考慮將輸出轉換為 `/dev/null` 或擷取為 `bash` 或 PowerShell 變數等方法來抑制命令輸出。
- 考慮存取您的日誌，並根據您的使用案例適當調整存取範圍。

Tools for PowerShell cmdlet 參考

Tools for PowerShell 提供 cmdlet，您可以用來存取 AWS 服務。若要查看有哪些 cmdlet 可用，請參閱 [AWS Tools for PowerShell cmdlet 參考](#)。

文件歷史紀錄

本主題說明 AWS Tools for PowerShell 文件的重大變更。

我們也會定期更新文件以回應客戶的意見回饋。若要傳送有關主題的意見回饋，請使用位於每個頁面底部的「此頁面是否有幫助？」旁邊的意見回饋按鈕。

如需變更和更新的其他資訊 AWS Tools for PowerShell，請參閱[版本備註](#)。如需獲得此文件更新的通知，您可以訂閱 [RSS 摘要](#)。

變更	描述	日期
最新消息	第 5 版 (V5) AWS Tools for PowerShell 已全面推出！如需詳細資訊，請參閱 AWS Tools for PowerShell 使用者指南 (V5) ，特別是 遷移至 V5 的主題。	2025 年 6 月 23 日
新功能	已發行的 V5 預覽內容 AWS Tools for PowerShell。	2025 年 6 月 13 日
新功能	已發佈 第 5 版（預覽版）的 使用者指南 AWS Tools for PowerShell。	2025 年 5 月 28 日
可觀測性	已宣布 GA 版本以提供可觀測性	2025 年 2 月 10 日
最新消息	新增有關完整性保護新預設行為的資訊。	2025 年 1 月 15 日
最新消息	新增 AWS Tools for PowerShell 第 5 版第一個預覽版本的相關資訊。	2024 年 11 月 18 日
管道、輸出和反覆運算	取代有關 \$AWSHistory 的內容，已棄用。	2024 年 10 月 10 日

可觀測性	新增 中可觀測性的預覽資訊 AWS Tools for PowerShell , 可收集遙測資料。	2024 年 9 月 13 日
在 Windows AWS Tools for PowerShell 上安裝	新增在擷取內容之前解除封鎖 ZIP 檔案的相關資訊。	2024 年 8 月 5 日
EC2-Classic 的相關資訊	已移除已淘汰的 EC2-Classic 相關資訊。	2024 年 8 月 1 日
程式碼範例	包含含有 cmdlet 範例的章節。	2024 年 4 月 17 日
其他安全性考量事項	包含有關敏感資料潛在記錄的 資訊。	2024 年 4 月 16 日
使用 設定工具身分驗證 AWS	新增 中 SSO 支援的相關資訊 AWS Tools for PowerShell。	2024 年 3 月 15 日
Tools for PowerShell cmdlet 參考	已新增區段，其中包含 Tools for PowerShell cmdlet 參考資 料的連結。	2023 年 11 月 17 日
納入了更多 IAM 最佳實務更新	更新了指南以符合 IAM 最佳實 務。如需更多詳細資訊，請參 閱 IAM 中的安全最佳實務 。	2023 年 10 月 12 日
在 Windows 上安裝	已移除有關使用已棄用 MSI 安 裝 Windows PowerShell 工具 的相關資訊。	2023 年 9 月 25 日
IAM 最佳實務更新	更新了指南以符合 IAM 最佳實 務。如需更多詳細資訊，請參 閱 IAM 中的安全最佳實務 。	2023 年 9 月 8 日
管道傳輸和 \$AWSHistory	已將 IncludeSensitiveDa ta 參數新增至 Set-AWSHi storyConfiguration Cmdlet。	2023 年 3 月 9 日

在 cmdlets 中使用 ClientConfig 參數	新增有關 ClientConfig 參數支援的資訊。	2022 年 10 月 28 日
使用 Windows PowerShell 啟動 Amazon EC2 執行個體	新增有關淘汰 EC2-Classical 的注意事項。	2022 年 7 月 26 日
AWS Tools for PowerShell 第 4 版	新增第 4 版的相關資訊，包括 Windows 和 Linux/macOS 的安裝說明，以及說明與第 3 版之差異及介紹新功能的 遷移 主題。	2019 年 11 月 21 日
AWS Tools for PowerShell 3.3.563	新增如何安裝及使用 AWS.Tools.Common 模組預覽版的相關資訊。這個新模組會將較舊的單體套件分成一個共用模組，每個 AWS 服務各一個模組。	2019 年 10 月 18 日
AWS Tools for PowerShell 3.3.343.0	已將資訊新增至 一節 AWS Tools for PowerShell ，介紹適用於 PowerShell Core AWS Lambda 的 Tools for PowerShell 開發人員以建置 AWS Lambda 函數。	2018 年 9 月 11 日
AWS Tools for Windows PowerShell 3.1.31.0	在 入門 章節中增加資訊，說明使用「安全性聲明標記語言」(SAML) 來支援設定使用者聯合身分的新 cmdlet。	2015 年 12 月 1 日
AWS Tools for Windows PowerShell 2.3.19	在 Cmdlet 探索和別名 區段中新增了有關新 Get-AWSCmdletName Cmdlet 的資訊，可協助使用者更輕鬆地找到所需的 AWS Cmdlet。	2015 年 2 月 5 日

[AWS Tools for Windows PowerShell 1.1.1.0](#)

2013 年 5 月 15 日

來自 Cmdlet 的集合輸出一律會列舉到 PowerShell 管道。自動支援可分頁的服務呼叫。新的 \$AWSHistory shell 變數會收集服務回應及選用的服務請求。AWSRegion 執行個體使用 Region 欄位 (而非 SystemName) 以利管道傳輸。Remove-S3Bucket 支援 -DeleteObjects 切換選項。修復 Set-AWSCredentials 的可用性問題。Initialize-AWSDefaults 會從其取得憑證和區域資料的位置報告。Stop-EC2Instance 接受 Amazon.EC2.Model.Reservation 執行個體做為輸入。泛型 List<T> 參數類型更換為陣列類型 (T[])。刪除或終止資源的 Cmdlet 會在刪除之前提示確認。Write-S3Object 支援將內嵌文字內容上傳至 Amazon S3。

[AWS Tools for Windows PowerShell 1.0.1.0](#)

Tools for Windows PowerShell 2012 年 12 月 21 日

PowerShell 模組的安裝位置已變更，因此使用 Windows PowerShell 第 3 版的環境可以利用自動載入。模組和支援檔案現在安裝到 AWS ToolsPowerShell 下方的 AWSPowerShell 子資料夾。安裝程式會自動移除在 AWS ToolsPowerShell 資料夾中，來自舊版的檔案。此版本中適用於 Windows PowerShell (所有版本) 的 PSModulePath 已更新為包含模組的上層資料夾 (AWS ToolsPowerShell)。

針對採用 Windows PowerShell 第 2 版的系統，Start (開始) 選單捷徑已更新為從新位置匯入模組，然後執行 Initialize-AWSDefaults。針對採用 Windows PowerShell 第 3 版的系統，Start (開始) 選單捷徑已更新為移除 Import-Module 命令，只留下 Initialize-AWSDefaults。如果您曾編輯 PowerShell 描述檔來執行 Import-Module 檔案的 AWSPowerShell.psd1，您需要將其更新以指向檔案的新位置 (如果使用 PowerShell 版本 3，則請移除 Import-Module 陳述式，因為已不再需要它)。由於這些變更，現在執行 Get-Module -ListAvailable 時，Tools

for Windows PowerShell 模組會列為可用模組。此外，對於 Windows PowerShell 版本 3 的使用者，執行由模組匯出的任何 cmdlet 都會在目前 PowerShell shell 中自動載入模組，不需要使用 Import-Module。這可在具有不允許指令碼執行之執行政策的系統上，啟用 cmdlet 互動用途。

[AWS Tools for Windows PowerShell 1.0.0.0](#)

初始版本

2012 年 12 月 6 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。