使用者指南

AWS CodeDeploy

API 版本 2014-10-06

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

AWS CodeDeploy: 使用者指南

Copyright © 2024 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

什麼是 CodeDeploy？

CodeDeploy 是一種部署服務，可自動將應用程式部署到 Amazon EC2 執行個體、現場部署執行個體、無伺服器 Lambda 函數或 Amazon ECS 服務。

您可以部署幾乎無限制的各種應用程式內容，包括：

- 代碼
- 無伺服器功能 AWS Lambda
- Web 與組態檔案
- Executables
- 套件
- 指令碼
- 多媒體檔案

CodeDeploy 可以部署在伺服器上執行並存放在 Amazon S3 儲存貯體、儲存庫或 Bitbucket 儲存 GitHub 庫中的應用程式內容。 CodeDeploy 也可以部署無伺服器 Lambda 函數。在使用之前，您不需要對現有程式碼進行變更 CodeDeploy。

CodeDeploy 讓您更容易：

- 快速推出新的功能。
- 更新 AWS Lambda 函數版本。
- 避免在部署應用程式時停機。
- 處理複雜的應用程式更新，而不會有許多與容易出錯的手動部署相關的風險。

服務能和您的基礎設施一同擴展，可以輕鬆部署至一個執行個體，也可以部署至數千個。

CodeDeploy 可搭配各種系統進行組態管理、原始檔控制、持續整合、持續交付和持續部署。如需詳細資訊，請參閱產品整合。

主 CodeDeploy 控台也提供快速搜尋資源的方法，例如儲存庫、建置專案、部署應用程式和管道。選擇 Go to resource (移至資源)，或按 / 鍵，然後輸入資源名稱。任何相符項目都會出現在清單中。搜尋不區分大小寫。您只會看到您有權檢視的資源。如需詳細資訊，請參閱 適用於 AWS CodeDeploy的 Identity and Access Management。

主題

的好處 AWS CodeDeploy

CodeDeploy 提供以下好處：

- 伺服器、無伺服器和容器應用程式。 CodeDeploy可讓您在部署無伺服器 AWS Lambda 功能版本或 Amazon ECS 應用程式的伺服器和應用程式上部署傳統應用程式。
- 自動化部署。 CodeDeploy 在您的開發、測試和生產環境中，完全自動化您的應用程式部署。CodeDeploy隨著您的基礎架構擴展，以便您可以部署到一個或數千個執行個體。
- 減少停機時間。如果您的應用程式使用 EC2 /內部部署運算平台， CodeDeploy 可協助您將應用程式的可用性 在就地部署期間，跨 Amazon EC2 CodeDeploy 執行個體執行滾動式更新。您可以指定一次要在離線時進行更新的執行個體數。在藍/綠部署期間，最新的應用程式修訂版已安裝在替代執行個體上。流量會依您的選擇，立刻或測試完新環境後，重新路由到這些執行個體。針對這兩種部署類型，根據您設定的規則 CodeDeploy追蹤應用程式健全狀況。
- 停止和復原。如有錯誤，您可以自動或手動停止和回復部署。
- 集中化控制。您可以透過 CodeDeploy 主控台或啟動和追蹤部署狀態 AWS CLI。您會收到一份報告，列出每個應用程式修訂版的部署時間以及 Amazon EC2 執行個體的部署時間。
- 易於採用。 CodeDeploy 與平台無關，可與任何應用程序一起使用。您可以輕鬆地重複使用設置代碼。 CodeDeploy 也可以與您的軟體發行程序或持續交付工具鏈整合。
- 並行部署。如果您有多個使用 EC2 /內部部署計算平台的應用程式， CodeDeploy 可以將它們同時部署到同一組執行個體。

CodeDeploy 運算平台概觀

CodeDeploy 能夠將應用程式部署到三個運算平台：

- EC2/ 現場部署：描述可以是 Amazon EC2 雲端執行個體、現場部署伺服器或兩者的實體伺服器執行個體。使用 EC2/ 內部部署運算平台建立的應用程式可由可執行檔、組態檔、映像等組成。

  使用 EC2 /內部部署計算平台的部署會使用就地或藍/綠部署類型來管理流量導向至執行個體的方式。如需詳細資訊，請參閱 CodeDeploy 部署類型概觀。

- AWS Lambda：用於部署包含 Lambda 函數的更新版本的應用程式。 AWS Lambda 在由高可用性運算結構組成的無伺服器運算環境中管理 Lambda 函數。計算資源的所有管理都由執行 AWS Lambda。如需詳細資訊，請參閱無伺服器運算與應用程式。如需 AWS Lambda 和 Lambda 函數的詳細資訊，請參閱AWS Lambda。

  您可以選擇初期測試、線性或組態，在部署期間管理流量轉移至更新 Lambda 函數版本的方 all-at-once 式。

- Amazon ECS：用於將 Amazon ECS 容器化應用程式部署為任務集。 CodeDeploy 將應用程式的更新版本安裝為新的取代工作集，以執行藍/綠部署。 CodeDeploy 將原始應用程式工作集的生產流量重新路由傳送至取代工作集。成功部署後，原始任務集會終止。有關 Amazon ECS 的更多信息，請參閱 Amazon 彈性容器服務。

  您可以選擇初期測試、線性或規劃，來管理部署期間流量轉移至更新工作集的方 all-at-once 式。

> **Note**
>
> 使用和 CodeDeploy 兩者都支援 Amazon ECS 藍/綠部署。 AWS CloudFormation後續各節將說明這些部署的詳細資訊。

下表說明 CodeDeploy 元件如何搭配每個運算平台使用。如需詳細資訊，請參閱：

- 使用中的部署群組 CodeDeploy
- 使用中的部署 CodeDeploy
- 使用中的部署組態 CodeDeploy
- 使用的應用程式修訂 CodeDeploy
- 使用中的應用程式 CodeDeploy

CodeDeploy 元件	EC2/內部部署	AWS Lambda	Amazon ECS
部署群組	部署修訂版到一組執行個體。	在高可用性運算基礎架構上部署新版本的無伺服器 Lambda 函數。	指定包含要部署為任務集的容器化應用程式的 Amazon ECS 服務、用於向已部署應用程式提供流量的生產和選用測試接聽程式、重新路由傳送流量和終止已部署應用程式的原始任務集的時機，以及選用的觸發器、警示和復原設定。
部署	部署由應用程式和 AppSpec 檔案組成的新修訂版本。 AppSpec 指定如何將應用程式部署到部署群組中的執行個體。	將生產流量從一個版本的 Lambda 函數轉換為相同函數的新版本。此 AppSpec 檔案會指定要部署的 Lambda 函數版本。	將 Amazon ECS 容器化應用程式的更新版本部署為新的替代任務集。CodeDeploy 將生產流量從具有原始版本的任

CodeDeploy 元件	EC2/內部部署	AWS Lambda	Amazon ECS
			務集重新路由到具有更新版本的新取代任務集。部署完成時，原始任務設定將終止。
部署組態	決定部署速度和在部署期間的任何點都必須健康的最少執行個體數的設定。	決定流量如何轉移至更新的 Lambda 函數版本的設定。	決定流量如何轉移至更新後的 Amazon ECS 任務集的設定。

CodeDeploy 元件	EC2/內部部署	AWS Lambda	Amazon ECS
修訂	AppSpec 檔案和應用程式檔案 (例如可執行檔、組態檔等) 的組合。	此 AppSpec 檔案可指定要部署哪些 Lambda 函數，以及可在部署生命週期事件掛接期間執行驗證測試的 Lambda 函數。	指定下列項目的 AppSpec 檔案： • Amazon ECS 服務的 Amazon ECS 任務定義，以及要部署的容器化應用程式。 • 您更新已部署應用程式的容器。 • 生產流量重新路由的容器連接埠。 • 選用的網路組態設定和 Lambda 函數，可在部署生命週期事件掛接期間執行驗證測試。

CodeDeploy 元件	EC2/內部部署	AWS Lambda	Amazon ECS
應用程式	部署群組和修訂版的集合。EC2/ 內部部署應用程式使用 EC2 /內部部署計算平台。	部署群組和修訂版的集合。用於 AWS Lambda 部署的應用程式會使用無伺服器 AWS Lambda 運算平台。	部署群組和修訂版的集合。用於 Amazon ECS 部署的應用程式使用 Amazon ECS 運算平台。

CodeDeploy 部署類型概觀

CodeDeploy 提供兩種部署類型選項：

- 就地部署：停止部署群組中每個執行個體上的應用程式、安裝最新的應用程式修訂版本，並啟動和驗證新版本的應用程式。您可以使用負載平衡器，以便在部署期間取消註冊每個執行個體，然後在部署完成後還原至服務。只有使用 EC2 /內部部署計算平台的部署才能使用就地部署。如需就地部署的更多資訊，請參閱就地部署概述。

  > **Note**
  >
  > AWS Lambda 和 Amazon ECS 部署無法使用就地部署類型。

- 藍/綠部署：部署的行為取決於您使用的運算平台：
  - EC2 /內部部署計算平台上的藍色/綠色：部署群組 (原始環境) 中的執行個體會使用下列步驟取代為不同的一組執行個體 (取代環境)：
    - 針對替代環境佈建執行個體。
    - 最新的應用程式修訂版會安裝在取代執行個體上。
    - 應用程式測試和系統驗證等活動會發生選擇性的等待時間。
    - 取代環境中的執行個體會使用一或多個 Elastic Load Balancing 負載平衡器登錄，導致流量重新路由傳送到這些執行個體。原始環境中的執行個體會取消註冊，並可終止或繼續執行以供其他用途使用。

> **Note**
>
> 如果您使用 EC2 /內部部署運算平台，請注意藍/綠部署僅適用於 Amazon EC2 執行個體。

- AWS Lambda 或 Amazon ECS 運算平台上的藍/綠：流量會根據初期測試、線性或all-at-once部署組態以遞增方式移動。
- 透過藍/綠部署 AWS CloudFormation：流量會從您目前的資源轉移到更新的資源，做為 AWS CloudFormation 堆疊更新的一部分。目前僅支援 ECS 藍/綠部署。

如需藍/綠部署的詳細資訊，請參閱 藍/綠部署概述。

> **Note**
>
> 您可以使用 CodeDeploy 代理程式在登入的執行個體上執行部署，而不需要應用程式、部署群組甚至是 AWS 帳戶。如需相關資訊，請參閱使用 CodeDeploy 代理程式驗證本機電腦上的部署套件。

主題

就地部署概述

> **Note**
>
> AWS Lambda 和 Amazon ECS 部署無法使用就地部署類型。

以下是就地部署的運作方式：

1. 首先，您可以在本機開發電腦或類似環境中建立可部署的內容，然後新增應用程式規格檔案 (AppSpec 檔案)。檔 AppSpec 案是唯一的 CodeDeploy。它會定義您要 CodeDeploy 執行的部署動作。您可以將可部署的內容和 AppSpec 檔案捆綁到存檔檔案中，然後將其上傳到 Amazon S3 儲存貯體或 GitHub存放庫。此封存檔稱為應用程式修訂版 (會只是修訂版)。

2. 接下來，您會提供 CodeDeploy 有關部署的資訊，例如要從哪個 Amazon S3 儲存貯體或 GitHub 儲存庫提取修訂版本，以及要將其內容部署到哪一組 Amazon EC2 執行個體。 CodeDeploy 將一組 Amazon EC2 執行個體呼叫為一個部署群組。部署群組包含個別標記的 Amazon EC2 執行個體、Amazon EC2 Auto Scaling 群組中的 Amazon EC2 執行個體，或兩者皆包含。

   每次成功上傳您要部署至部署群組的新應用程式修訂版時，該綁定會設定為部署群組的目標修訂版。換言之，目前鎖定為部署目標的應用程式修訂版就是目標修訂版。這也是會拿來自動部署的修訂版。
3. 接下來，每個執行個體上的 CodeDeploy 代理程式都會輪詢 CodeDeploy 以決定從指定的 Amazon S3 儲存貯體或 GitHub 儲存庫提取的內容和時機。
4. 最後，每個執行個體上的 CodeDeploy 代理程式會從 Amazon S3 儲存貯體或 GitHub 儲存庫提取目標修訂，然後使用 AppSpec 檔案中的指示將內容部署到執行個體。

CodeDeploy 保留部署記錄，以便您取得部署狀態、部署組態參數、執行個體健全狀況等。

藍/綠部署概述

藍/綠部署可用來更新應用程式，同時將新應用程式版本變更所造成的中斷降到最低。 CodeDeploy 在重新路由您的生產流量之前，將新的應用程式版本與舊版本一起佈建。

- AWS Lambda：流量從 Lambda 函數的一個版本轉移到相同 Lambda 函數的新版本。
- Amazon ECS：流量從 Amazon ECS 服務中設置的任務轉移到相同 Amazon ECS 服務中設置的更新替換任務。
- EC2 /內部部署：流量從原始環境中的一組執行個體轉移到一組取代的執行個體。

所有 AWS Lambda 和 Amazon ECS 部署均為藍/綠。EC2 /內部部署可以就地或藍色/綠色。藍/綠部署比現場部署多出許多優勢：

- 您可以在新的取代環境中安裝和測試應用程式，並透過重新路由流量，將其部署至生產環境。
- 如果您使用的是 EC2 /內部部署計算平台，切換回最新版本的應用程式會更快、更可靠。因為流量只要尚未終止，都可以路由回原始執行個體。使用現場部署，版本必須復原，方法是重新部署之前版本的應用程式。
- 如果您使用的是 EC2 /內部部署計算平台，則會為藍/綠部署佈建新的執行個體，並反映出大 up-to-date 部分的伺服器組態。這可協助您避免有時在長時間執行的執行個體上發生的問題類型。
- 如果您使用的是 AWS Lambda 運算平台，您可以控制流量從原始 AWS Lambda 函數版本轉移到新的 AWS Lambda 函數版本的方式。

- 如果您使用 Amazon ECS 運算平台，則可以控制流量從原始任務集轉移到新任務集的方式。

藍/綠部署 AWS CloudFormation 可以使用下列其中一種方法：

- AWS CloudFormation 部署範本：使用 AWS CloudFormation 範本規劃部署時，您的部署會由 AWS CloudFormation 更新觸發。當您變更資源並上傳範本變更時，中的堆疊更新 AWS CloudFormation 會起始新部署。如需可在 AWS CloudFormation 範本中使用的資源清單，請參閱AWS CloudFormation CodeDeploy供參考的範本。
- 藍/綠部署透過 AWS CloudFormation：您可以使用透過堆疊更新 AWS CloudFormation 來管理藍/綠部署。除了在堆疊範本中指定流量路由和穩定設定之外，您還可以定義藍色和綠色資源。然後，如果您在堆疊更新期間更新選取的資源， AWS CloudFormation 會產生所有必要的綠色資源，根據指定的流量路由參數轉移流量，然後刪除藍色資源。如需詳細資訊，請參閱使用AWS CloudFormation者指南 AWS CloudFormation中的 CodeDeploy 使用自動化 Amazon ECS 藍/綠部署。

> **Note**
>
> 僅支援 Amazon ECS 藍色/綠色部署。

您如何設定藍/綠部署取決於您部署所使用的運算平台。

在 AWS Lambda 或Amazon ECS 運算平台上的藍色/綠色部署

如果您使用 AWS Lambda 或 Amazon ECS 運算平台，則必須指出流量如何從原始 AWS Lambda 功能或 Amazon ECS 任務集轉移到新功能或任務集。若要指出流量的轉移方式，您必須指定下列其中一個部署組態：

- 金絲雀
- 線性
- all-at-once

如需有關在初期測試、線性或 all-at-once部署規劃中流量如何轉移的資訊，請參閱部署組態。

如需 Lambda 部署組態的詳細資訊，請參閱AWS Lambda 運算平台上的部署組態。

如需 Amazon ECS 部署組態的詳細資訊，請參閱在 Amazon ECS 運算平台上的部署組態。

在EC2/內部署計算平台上的藍色/綠色部署

> **Note**
>
> 您必須使用 Amazon EC2 執行個體在EC2/ 現場部署運算平台上進行藍/綠部署。藍/綠部署類型不支援內部部署執行個體。

如果您使用的是 EC2 /內部部署計算平台，則適用以下條件：

您必須擁有一個或多個具有識別 Amazon EC2 標籤或 Amazon EC2 Auto Scaling 群組的 Amazon EC2 執行個體。執行個體必須符合下列額外的要求：

- 每個 Amazon EC2 執行個體都必須附加正確的 IAM 執行個體設定檔。
- CodeDeploy 代理程式必須安裝並在每個執行個體上執行。

> **Note**
>
> 您通常也可以在您的原始環境中的執行個體上執行應用程式修訂版，但這不是藍/綠部署的要求。

當您建立用於藍/綠部署的部署群組時，您可以選擇如何指定您的替換環境：

複製現有的 Amazon EC2 Auto Scaling 群組：在藍/綠部署期間， CodeDeploy 會在部署期間為替換環境建立執行個體。使用此選項，可 CodeDeploy 使用您指定的 Amazon EC2 Auto Scaling 群組做為替代環境的範本，包括相同數量的執行中執行個體和許多其他組態選項。

手動選擇執行個體：您可以使用 Amazon EC2 執行個體標籤、Amazon EC2 Auto Scaling 群組名稱或兩者，指定要計為替代的執行個體。若您選擇此選項，直到建立部署前，您都無需指定取代環境的執行個體。

以下是其運作方式：

1. 您已經有執行個體或做為您原始環境使用的 Amazon EC2 Auto Scaling 群組。第一次執行藍色/綠色部署時，通常會使用已在就地部署中使用的執行個體。
2. 在現有的 CodeDeploy 應用程式中，您可以建立藍/綠部署群組，其中除了就地部署所需的選項之外，還可以指定下列項目：

- 在藍/綠部署程序期間，可將流量從原始環境路由至替換環境的負載平衡器或負載平衡器。
- 是否要立即將流量重新路由至取代環境，或是等待您手動重新路由流量。
- 將流量路由至取代執行個體的速度。
- 是否終止或讓遭取代的執行個體繼續執行。

3. 您會建立此部署群組的部署，期間會發生下列事件：
   a. 如果您選擇複製 Amazon EC2 Auto Scaling 群組，則會為您的替代環境佈建執行個體。
   b. 您為部署指定的應用程式修訂會在取代執行個體上安裝。
   c. 若您在部署群組設定中指定等待時間，部署便會暫停。您可以在此時於您的取代環境中執行測試及驗證。若您沒有在等待期間結束前手動重新路由流量，部署便會停止。
   d. 取代環境中的執行個體會向 Elastic Load Balancing 器登錄，流量開始路由到這些執行個體。
   e. 原始環境中的執行個體會取消註冊，並會根據您在部署群組中指定的內容處理 (終止或持續執行)。

藍/綠部署 AWS CloudFormation

您可以使用範本建立資源 AWS CloudFormation 模型，以管理 CodeDeploy 藍/綠部署。

當您使用 AWS CloudFormation 範本建立藍/綠資源的模型時，您可以在中建立堆疊更新， AWS CloudFormation 以更新您的工作集。生產流量會從服務的原始任務集轉移到替代任務集，可以一次全部、線性部署和封裝時間，或是使用 canary 部署。堆疊更新會啟動中 CodeDeploy的部署。您可以在中檢視部署狀態和歷程記錄 CodeDeploy，但不會在 AWS CloudFormation 範本之外建立或管理 CodeDeploy 資源。

> **Note**
>
> 對於透過藍/綠部署 AWS CloudFormation，您不會建立 CodeDeploy 應用程式或部署群組。

此方法僅支援 Amazon ECS 藍色/綠色部署。如需透過藍/綠部署的更多資訊 AWS CloudFormation，請參閱透過以下方式建立亞馬遜ECS 藍色/綠色部署 AWS CloudFormation。

我們希望傾聽您的意見

我們誠摯歡迎您提供意見回饋。要聯繫我們，請訪問 CodeDeploy 論壇。

主題

CodeDeploy 主要元件

在開始使用服務之前，您應該先熟悉 CodeDeploy 部署程序的主要元件。

主題

應用程式

應用程式是唯一識別您要部署的應用程式的名稱。 CodeDeploy 使用此名稱做為容器使用，以確保在部署期間參考修訂版本、部署組態和部署群組的正確組合。

運算平台

運算平台是 CodeDeploy 部署應用程式的平台。共有三種運算平台：

- EC2/ 現場部署：描述可以是 Amazon EC2 雲端執行個體、現場部署伺服器或兩者的實體伺服器執行個體。使用 EC2/ 內部部署運算平台建立的應用程式可由可執行檔、組態檔、映像等組成。

  使用 EC2 /內部部署計算平台的部署會使用就地或藍/綠部署類型來管理流量導向至執行個體的方式。如需詳細資訊，請參閱 CodeDeploy 部署類型概觀。

- AWS Lambda：用於部署包含 Lambda 函數的更新版本的應用程式。 AWS Lambda 在由高可用性運算結構組成的無伺服器運算環境中管理 Lambda 函數。計算資源的所有管理都由執行 AWS Lambda。如需詳細資訊，請參閱無伺服器運算與應用程式。如需 AWS Lambda 和 Lambda 函數的詳細資訊，請參閱AWS Lambda。

  您可以選擇初期測試、線性或組態，在部署期間管理流量轉移至更新 Lambda 函數版本的方 all-at-once 式。
- Amazon ECS：用於將 Amazon ECS 容器化應用程式部署為任務集。 CodeDeploy 將應用程式的更新版本安裝為新的取代工作集，以執行藍/綠部署。 CodeDeploy 將原始應用程式工作集的生產流量重新路由傳送至取代工作集。成功部署後，原始任務集會終止。有關 Amazon ECS 的更多信息，請參閱 Amazon 彈性容器服務。

  您可以選擇初期測試、線性或規劃，來管理部署期間流量轉移至更新工作集的方 all-at-once 式。

> **Note**
>
> 透過和 CodeDeploy 兩者都支援 Amazon ECS 藍/綠部署。 AWS CloudFormation後續各節將說明這些部署的詳細資訊。

部署組態

部署組態是一組部署規則，以及部署成功和失敗條件的部署在部署 CodeDeploy 期間使用。如果您的部署使用 EC2 /內部部署計算平台，您可以指定部署運作狀態良好的執行個體數目下限。如果您的部署使用 AWS Lambda 或 Amazon ECS 運算平台，您可以指定如何將流量路由到更新的 Lambda 函數或 ECS 任務集。

如需針對使用 EC2 /內部部署計算平台之部署指定正常狀態良好的主機數目下限的詳細資訊，請參閱。關於健全狀態執行個體的最小數目

下列部署組態會指定在使用 Lambda 或 ECS 運算平台的部署期間如何路由流量：

- Canary：流量以兩個增量轉移。您可以從預先定義的初期測試選項中進行選擇，這些選項指定在第一個增量中轉移至更新 Lambda 函數或 ECS 任務集的流量百分比，以及在第二個增量中移動剩餘流量之前的間隔 (以分鐘為單位)。
- Linear (線性)：流量以每個增量之間的相等分鐘數以同等增量轉移。您可從預先指定的線性選項中指定每次增量的流量轉移百分比例，以及在每個增量之間的分鐘數。
- ll-at-once答：所有流量都會一次從原始 Lambda 函數或 ECS 任務集轉移到更新的函數或任務集。

部署群組

部署群組是一組個別執行個體。部署群組包含個別標記的執行個體、Amazon EC2 Auto Scaling 群組中的 Amazon EC2 執行個體，或兩者皆包含。如需 Amazon EC2 執行個體標籤的相關資訊，請參閱使用主控台使用標籤。如需內部部署執行個體的資訊，請參閱「Working with On-Premises Instances」。如需 Amazon EC2 Auto Scaling 的相關資訊，請參閱 CodeDeploy 與 Amazon EC2 Auto Scaling 集成。

部署類型

部署類型是一種方法，用來在部署群組中的執行個體上提供最新的應用程式修訂版本。有兩個部署類型：

- 就地部署：停止部署群組中每個執行個體上的應用程式、安裝最新的應用程式修訂版本，並啟動和驗證新版本的應用程式。您可以使用負載平衡器，以便在部署期間取消註冊每個執行個體，然後在部署完成後還原至服務。只有使用 EC2 /內部部署計算平台的部署才能使用就地部署。如需就地部署的更多資訊，請參閱就地部署概述。
- 藍/綠部署：部署的行為取決於您使用的運算平台：
  - EC2 /內部部署計算平台上的藍色/綠色：部署群組 (原始環境) 中的執行個體會使用下列步驟取代為不同的一組執行個體 (取代環境)：
    - 針對替代環境佈建執行個體。
    - 最新的應用程式修訂版會安裝在取代執行個體上。
    - 應用程式測試和系統驗證等活動會發生選擇性的等待時間。
    - 取代環境中的執行個體會使用一或多個 Elastic Load Balancing 負載平衡器登錄，導致流量重新路由傳送到這些執行個體。原始環境中的執行個體會取消註冊，並且可以終止或繼續執行以供其他用途使用。

    > **Note**
    >
    > 如果您使用 EC2 /內部部署運算平台，請注意藍/綠部署僅適用於 Amazon EC2 執行個體。

  - AWS Lambda 或 Amazon ECS 運算平台上的藍/綠：流量會根據初期測試、線性或all-at-once部署組態以遞增方式移動。
  - 透過藍/綠部署 AWS CloudFormation：流量會從您目前的資源轉移到更新的資源，做為 AWS CloudFormation 堆疊更新的一部分。目前僅支援 ECS 藍/綠部署。

  如需藍/綠部署的詳細資訊，請參閱 藍/綠部署概述。

> **Note**
>
> 使用和 CodeDeploy 兩者都支援 Amazon ECS 藍色/綠色部署。 AWS CloudFormation後續各節將說明這些部署的詳細資訊。

IAM 執行個體描述檔

IAM 執行個體設定檔是您附加到 Amazon EC2 執行個體的 IAM 角色。此設定檔包括存取存放應用程式的 Amazon S3 儲存貯體或存 GitHub 放庫所需的許可。如需詳細資訊，請參閱 步驟 4：為您的 Amazon EC2 執行個體建立 IAM 執行個體設定檔。

修訂

修訂版是應用程式的版本。 AWS Lambda 部署修訂版本是 YAML 或 JSON 格式的檔案，可指定要部署之 Lambda 函數的相關資訊。EC2 /內部部署修訂版本為包含來源內容 (原始程式碼、網頁、可執行檔和部署指令碼) 和應用程式規格檔案 (檔案) 的封存檔AppSpec 案。 AWS Lambda 修訂版本可存放在 Amazon S3 儲存貯體中。EC2 /內部部署修訂存儲在Amazon S3 儲存貯體或 GitHub 儲存庫中。對於 Amazon S3，修訂版本是由其 Amazon S3 物件金鑰及其 ETag、版本或兩者唯一識別的。對於 GitHub，修訂是由其提交 ID 唯一標識。

服務角色

服務角色是一種 IAM 角色，可授予 AWS 服務許可，以便它可以存取 AWS 資源。您附加至服務角色的原則會決定服務可存取哪些 AWS 資源，以及可以對這些資源執行的動作。對於 CodeDeploy，服務角色可用於下列項目：

- 讀取套用至執行個體的標籤或與執行個體相關聯的 Amazon EC2 Auto Scaling 群組名稱。這可 CodeDeploy 以識別它可以部署應用程式的執行個體。
- 在執行個體、Amazon EC2 Auto Scaling 群組和 Elastic Load Balancing 負載平衡器上執行操作。
- 將資訊發佈到 Amazon SNS 主題，以便在發生指定的部署或執行個體事件時傳送通知。
- 擷取 CloudWatch 警示的相關資訊，以針對部署設定警示監控。

如需詳細資訊，請參閱 步驟 2：建立服務角色 CodeDeploy。

目標修訂

目標修訂版是您已上傳至儲存庫並想要部署到部署群組中的執行個體的最新版本應用程式修訂版本。換言之，就是目前鎖定為部署目標的應用程式修訂版。這也是會拿來自動部署的修訂版。

其他組件

如需 CodeDeploy 工作流程中其他元件的相關資訊，請參閱下列主題：

- 選擇 CodeDeploy 存放庫類型
- Deployments
- Application Specification Files
- Instance Health
- 與 CodeDeploy 代理工作
- Working with On-Premises Instances

CodeDeploy 部署

本主題提供中部署的元件和工作流程的相關資訊 CodeDeploy。部署程序會根據您用於部署的運算平台或部署方法 (Lambda、Amazon ECS、EC2 或現場部署或通過 AWS CloudFormation) 而有所不同。

主題

- 在 AWS Lambda 運算平台上進行部署
- 在 Amazon ECS 運算平台上部署
- EC2/內部部署計算平台上的部署

在 AWS Lambda 運算平台上進行部署

本主題提供使用 AWS Lambda 運算平台之 CodeDeploy 部署元件和工作流程的相關資訊。

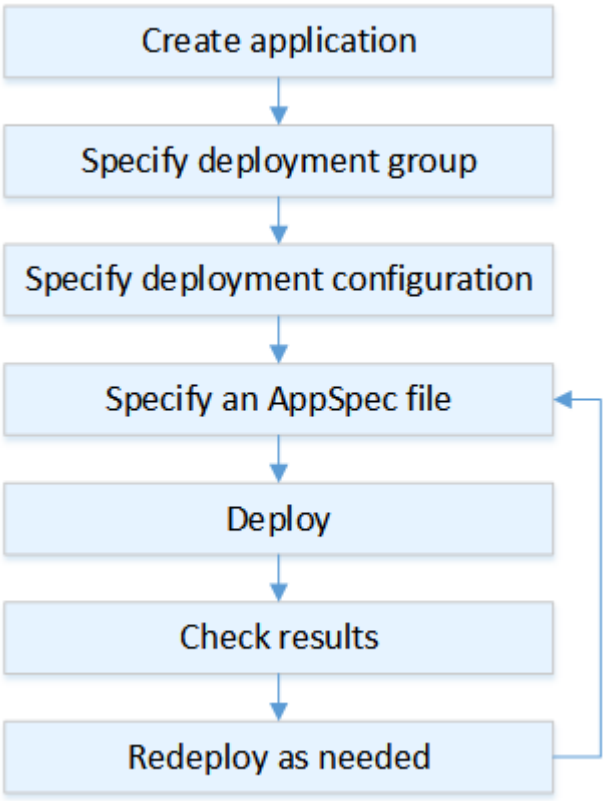
主題

- AWS Lambda 運算平台上的部署工作流程
- 上傳應用程式修訂
- 建立應用程式和部署群組
- 部署應用程式修訂

- 更新您的申請
- 已停止和失敗的部署
- 重新部署和部署復原

AWS Lambda 運算平台上的部署工作流程

下圖顯示新的及更新 AWS Lambda 函數部署中的主要步驟。



這些步驟包括：

1. 建立應用程式並為其命名，而其名稱需唯一識別您要部署的應用程式修訂。若要部署 Lambda 函數，請在建立應用程式時選擇 AWS Lambda 運算平台。 CodeDeploy 在部署期間使用此名稱，以確保其參考正確的部署元件，例如部署群組、部署組態和應用程式修訂。如需詳細資訊，請參閱 建立應用程式 CodeDeploy。
2. 指定部署群組的名稱來設定部署群組。
3. 選擇部署組態，以指定流量從原始 AWS Lambda 函數版本轉移到新 Lambda 函數版本的方式。如需詳細資訊，請參閱 View Deployment Configuration Details。

4. 將應用程式規格檔案 (AppSpec 檔案) 上傳到 Amazon S3。此 AppSpec 檔案會指定用來驗證部署的 Lambda 函數版本和 Lambda 函數。如果您不想建立 AppSpec 檔案，可以使用 YAML 或 JSON 直接在主控台中指定 Lambda 函數版本和 Lambda 部署驗證函數。如需詳細資訊，請參閱 使用的應用程式修訂 CodeDeploy。
5. 將應用程式修訂部署至部署群組。 AWS CodeDeploy 部署您指定的 Lambda 函數修訂。流量會使用您在建立應用程式時選擇的部署 AppSpec 檔案轉移至 Lambda 函數修訂版。如需詳細資訊，請參閱 使用建立部署 CodeDeploy。
6. 檢查部署結果。如需詳細資訊，請參閱 監控部署 CodeDeploy。

上傳應用程式修訂

將 AppSpec 檔案放置在 Amazon S3 中，或直接將檔案輸入主控台或 AWS CLI。如需詳細資訊，請參閱 Application Specification Files。

建立應用程式和部署群組

AWS Lambda 運算平台上的 CodeDeploy 部署群組可識別一或多個 AppSpec 檔案的集合。每個 AppSpec 檔案都可以部署一個 Lambda 函數版本。部署群組也會定義未來部署的一組組態選項 (例如警示和轉返組態)。

部署應用程式修訂

現在，您已準備好將 AppSpec 檔案中指定的函數修訂版部署到部署群組。您可以使用 CodeDeploy 控制台或創建部署命令。您可以指定多個參數來控制部署 (包含修訂、部署群組和部署組態)。

更新您的申請

您可以對應用程式進行更新，然後使用 CodeDeploy 主控台或呼叫 create-deployment 命令來推送修訂。

已停止和失敗的部署

您可以使用 CodeDeploy 控制台或停止部署命令來停止部署。當您嘗試停止部署時，會發生下列三者之一：

- 部署停止，而且操作傳回成功狀態。在此情況下，不需要在已停止部署的部署群組上執行其他部署生命週期事件。
- 此部署不會立即停止，而且操作傳回擱置中狀態。在此情況下，一些部署生命週期事件可能仍然在部署群組上執行。擱置中操作完成之後，後續呼叫停止部署會傳回成功狀態。

- 部署無法停止，而且操作傳回錯誤。如需詳細資訊，請參閱 AWS CodeDeploy API 參考中的ErrorInformation和常見錯誤。

失敗的部署可能導致已執行某些部署生命週期事件，就像已停止的部署一樣。若要瞭解部署失敗的原因，您可以使用 CodeDeploy主控台或分析失敗部署中的記錄檔資料。如需詳細資訊，請參閱 應用程式修訂和記錄檔清理 及 檢視 CodeDeploy EC2/內部部署的記錄資料。

重新部署和部署復原

CodeDeploy 透過將先前部署的修訂版重新部署 (做為新部署) 來實作復原。

您可以設定部署群組以在符合特定條件時自動轉返部署 (包含部署失敗或符合警示監控閾值時)。您也可以覆寫針對個別部署中部署群組所指定的轉返設定。

您也可以手動重新部署先前部署的修訂，以選擇轉返失敗部署。

在所有情況下，新的或轉返的部署會獲指派其專屬部署 ID。您可以在 CodeDeploy 主控台中檢視的部署清單會顯示哪些部署是自動部署的結果。

如需詳細資訊，請參閱 使用以下方式重新部署和復原部署 CodeDeploy。

在 Amazon ECS 運算平台上部署

本主題提供使用 Amazon ECS 運算平台之 CodeDeploy 部署的元件和工作流程的相關資訊。

主題

在您開始 Amazon ECS 部署之前

開始 Amazon ECS 應用程式部署之前，您必須準備好下列項目。某些需求是在您建立部署群組時指定的，而有些需求則會在 AppSpec 檔案中指定。

需求	指定位置
Amazon ECS 叢集	部署群組
Amazon ECS 服務	部署群組
Application Load Balancer 或 Network Load Balancer	部署群組
生產接聽程式	部署群組
測試接聽程式 (選用)	部署群組
兩個目標群組	部署群組
Amazon ECS 任務定義	AppSpec 檔案
容器名稱	AppSpec 檔案
容器連接埠	AppSpec 檔案

Amazon ECS 集群

Amazon ECS 叢集是任務或服務的邏輯分組。建立 CodeDeploy 應用程式的部署群組時，請指定包含 Amazon ECS 服務的 Amazon ECS 叢集。如需詳細資訊，請參閱 Amazon 彈性容器服務使用者指南中的 Amazon ECS 叢集。

Amazon ECS 服務

Amazon ECS 服務會在 Amazon ECS 叢集中維護和執行任務定義的指定執行個體。您的 Amazon ECS 服務必須啟用 CodeDeploy。根據預設，Amazon ECS 服務是針對 Amazon ECS 部署啟用的。建立部署群組時，您可以選擇在 Amazon ECS 叢集中部署 Amazon ECS 服務。如需詳細資訊，請參閱 Amazon 彈性容器服務使用者指南中的 Amazon ECS 服務。

Application Load Balancer 或 Network Load Balancer

您必須將 Elastic Load Balancing 與您想要透過 Amazon ECS 部署更新的 Amazon ECS 服務搭配使用。您可以使用應用程式負載平衡器或 Network Load Balancer。我們建議您使 Application Load Balancer，以利用動態連接埠對應、路徑型路由和優先順序規則等功能。您可以在建立 CodeDeploy 應用程式的部署群組時指定負載平衡器。如需詳細資訊，請參閱 Amazon 彈性容器服務使用者指南中的為 CodeDeploy Amazon ECS 部署設定負載平衡器、目標群組和接聽程式和建立負載平衡器。

一個或兩個聽眾

接聽程式供負載平衡器用來將流量導向到您的目標群組。一個生產接聽程式為必要項目。您可以指定選用的第二個測試接聽程式，負責在您執行驗證測試時引導流量到您的替換任務。當您建立部署群組時，您指定一或兩個接聽程式。如果您使用 Amazon ECS 主控台建立 Amazon ECS 服務，則會為您建立接聽程式。如需詳細資訊，請參閱 E lastic Load Balancing 使用者指南中的應用程式負載平衡器的接聽程式和 Amazon 彈性容器服務使用者指南中的建立服務。

兩個 Amazon ECS 目標群體

目標群組是用於將流量路由到已註冊的目標。Amazon ECS 部署需要兩個目標群組：一個用於 Amazon ECS 應用程式的原始任務集，另一個用於其替換任務集。在部署期間， CodeDeploy 會建立取代工作集，並將原始工作集的流量重新路由傳送至新工作集。您可以在建立 CodeDeploy 應用程式的部署群組時指定目標群組。

在部署期間， CodeDeploy 決定哪個目標群組與具有狀態 PRIMARY (這是原始任務集) 的 Amazon ECS 服務中的任務集相關聯，並將一個目標群組與其關聯，然後將另一個目標群組與替換任務集相關聯。如果您進行其他部署，與目前部署之原始任務集相關聯的目標群組會與下一個部署的替換任務集建立關聯。如需詳細資訊，請參閱 E lastic Load Balancing 使用者指南中的應用程式負載平衡器的目標群組。

Amazon ECS 任務定義

若要執行包含 Amazon ECS 應用程式的 Docker 容器，則需要任務定義。您可以在 CodeDeploy 應用程式的 AppSpec 檔案中指定工作定義的 ARN。如需詳細資訊，請參閱 Amazon 彈性容器服務使用者指南中的 Amazon ECS 任務定義和 AppSpec Amazon ECS 部署的「資源」部分。

適用於您的 Amazon ECS 應用程式的容器

Docker 容器是一個軟體單位，將程式碼和其相依性封裝，讓您的應用程式能夠執行。容器會隔離您的應用程式，讓其在不同的運算環境中執行。負載平衡器會將流量導向 Amazon ECS 應用程式任務集中的容器。您可以在 CodeDeploy 應用程序的 AppSpec 文件中指定容器的名稱。 AppSpec 檔案中指定的容器必須是 Amazon ECS 任務定義中指定的容器之一。如需詳細資訊，請參閱什麼是

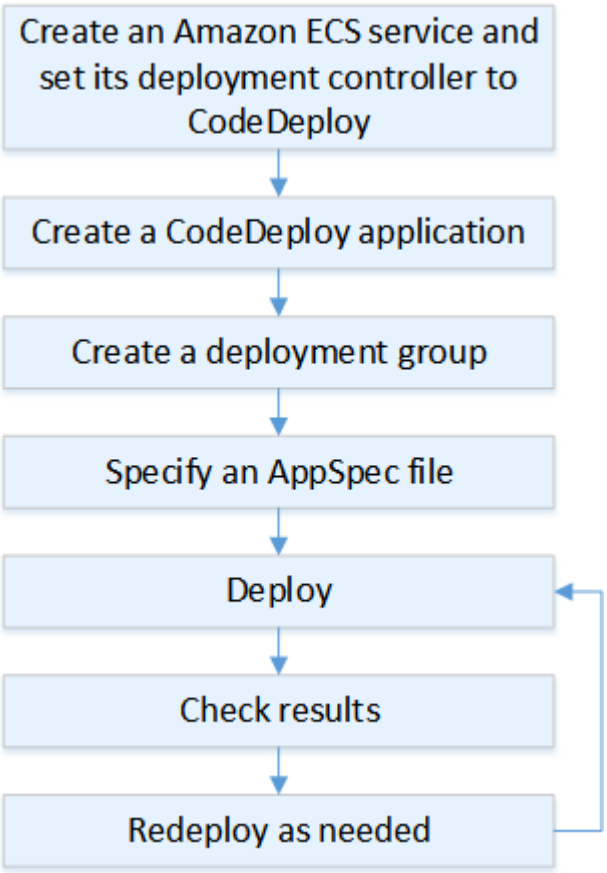
Amazon 彈性容器服務？ 在 Amazon 彈性容器服務用戶指南和 AppSpec Amazon ECS 部署的「資源」部分.

取代任務集的連接埠

在 Amazon ECS 部署期間，負載平衡器會將流量導向至 CodeDeploy應用程式 AppSpec 檔案中指定容器上的此連接埠。您可以在 CodeDeploy 應用程式的 AppSpec 檔案中指定連接埠。如需詳細資訊，請參閱 AppSpec Amazon ECS 部署的「資源」部分。

Amazon ECS 運算平台上的部署工作流程 (高階)

下圖顯示部署更新的 Amazon ECS 服務的主要步驟。



這些步驟包括：

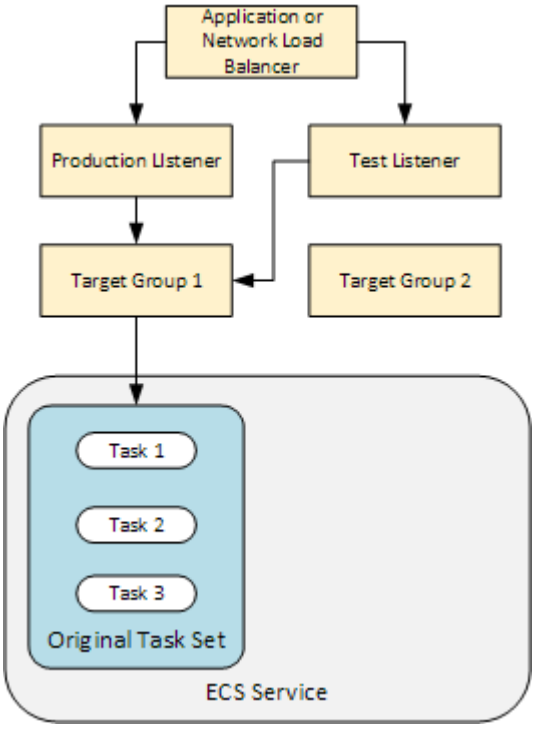
1. 指定唯一代表您要部署之內容的名稱，以建立 AWS CodeDeploy 應用程式。若要部署 Amazon ECS 應用程式，請在您的 AWS CodeDeploy 應用程式中選擇 Amazon ECS 運算平台。CodeDeploy 在部署期間使用應用程式來參考正確的部署元件，例如部署群組、目標群組、監聽器和流量重新路由傳送行為，以及應用程式修訂版本。如需詳細資訊，請參閱 建立應用程式 CodeDeploy。

2. 指定下列項目來設定部署群組：
   - 部署群組名稱。
   - 您的 Amazon ECS 叢集和服務名稱。Amazon ECS 服務的部署控制器必須設定為 CodeDeploy。
   - 生產接聽程式、選用的測試接聽程式和部署期間使用的目標群組。
   - 部署設定，例如何將生產流量重新路由到 Amazon ECS 服務中設定的替代 Amazon ECS 任務，以及何時終止 Amazon ECS 服務中設定的原始 Amazon ECS 任務。
   - 選用設定，例如觸發條件、警示和轉返行為。
3. 指定應用程式規格檔案 (AppSpec 檔案)。您可以將它上傳到 Amazon S3、以 YAML 或 JSON 格式將其輸入主控台，或使用 AWS CLI 或開發套件指定。該 AppSpec 檔案指定部署的 Amazon ECS 任務定義、用於路由流量的容器名稱和連接埠對應，以及 Lambda 函數在部署生命週期掛鉤後執行。容器名稱必須是 Amazon ECS 任務定義中的容器。如需詳細資訊，請參閱 使用的應用程式修訂 CodeDeploy。
4. 部署您的應用程式修訂。 AWS CodeDeploy 將流量從 Amazon ECS 服務中的原始版本任務集重新路由到新的替換任務集。部署群組中指定的目標群組是用於為原始和替換任務集提供流量。部署完成後，原始任務集會終止。您可以指定選用的測試接聽程式，在將流量重新路由到替換版本之前，為您的替換版本提供測試流量。如需詳細資訊，請參閱 使用建立部署 CodeDeploy。
5. 檢查部署結果。如需詳細資訊，請參閱 監控部署 CodeDeploy。

Amazon ECS 部署期間會發生什麼情況

在使用測試接聽程式啟動 Amazon ECS 部署之前，您必須先設定其元件。如需詳細資訊，請參閱 在您開始 Amazon ECS 部署之前。

下圖顯示當 Amazon ECS 部署準備就緒時，這些元件之間的關係。



部署開始，部署生命週期事件開始逐一執行。某些生命週期事件是只執行 AppSpec 檔案中指定的 Lambda 函數的掛接。下表中的部署生命週期事件依其執行順序列出。如需詳細資訊，請參閱 AppSpec Amazon ECS 部署的「掛鉤」部分。

週期事件	生命週期事件動
`BeforeInstall` （Lambda 函數的鉤子）	執行 Lambda 函數。
安裝	設定替換任務集。
`AfterInstall` （Lambda 函數的鉤子）	執行 Lambda 函數。
AllowTestTraffic	將流量從測試接聽程式路由到目標群組 2。
`AfterAllowTestTraffic` （Lambda 函數的鉤子）	執行 Lambda 函數。
`BeforeAllowTraffic` （Lambda 函數的鉤子）	執行 Lambda 函數。
AllowTraffic	將流量從生產接聽程式路由到目標群組 2。

週期事件	生命週期事件動
`AfterAllowTraffic`	執行 Lambda 函數。

> **Note**
>
> 鉤子中的 Lambda 函數是可選的。

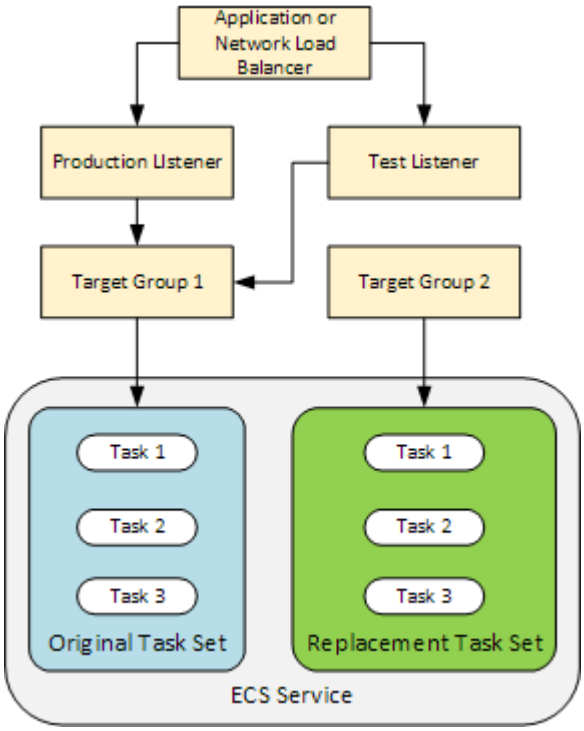
1. 執行檔案中`BeforeInstall`掛接中指定的任何 Lambda 函 AppSpec 數。
2. 在`Install` 生命週期事件期間：

   a. 取代任務集會在您的 Amazon ECS 服務中建立。

   b. 更新的容器化應用程式會安裝到替換任務集。

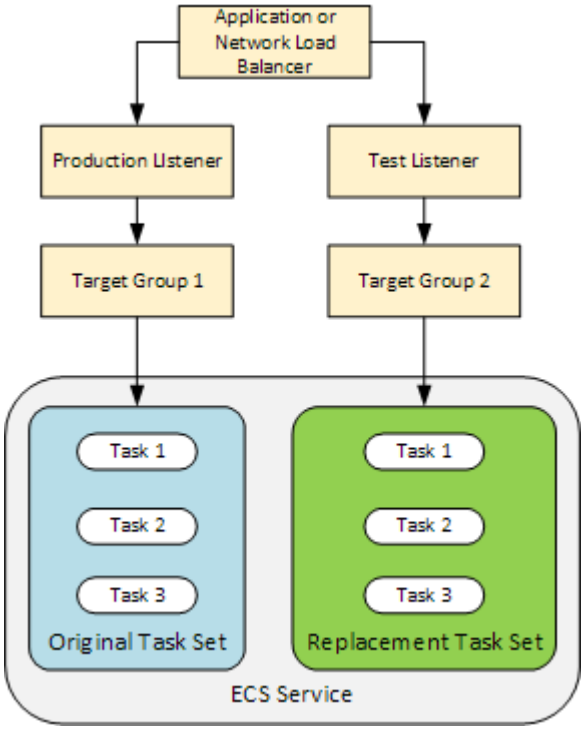
   c. 第二個目標群組與替換任務集相關聯。

   下圖顯示具有新替代任務集的部署元件。容器化應用程式在此任務集中。任務集包含三個任務。(應用程式可以有任何數量的任務。) 第二個目標群組現在與替換任務集相關聯。



3. 執行檔案中`AfterInstall`掛接中指定的任何 Lambda 函 AppSpec 數。

4. 已叫用 `AllowTestTraffic` 事件。在這個生命週期事件期間，測試接聽程式將流量路由到更新的容器化應用程式。



5. 執行檔案中`AfterAllowTestTraffic`掛接中指定的任何 Lambda 函 AppSpec 數。Lambda 函數可以使用測試流量來驗證部署。例如，Lambda 函數可以為測試接聽程式提供流量，以及追蹤來自替代任務集的指標。如果已設定復原，您可以設定 CloudWatch 警示，在 Lambda 函數中的驗證測試失敗時觸發回復。

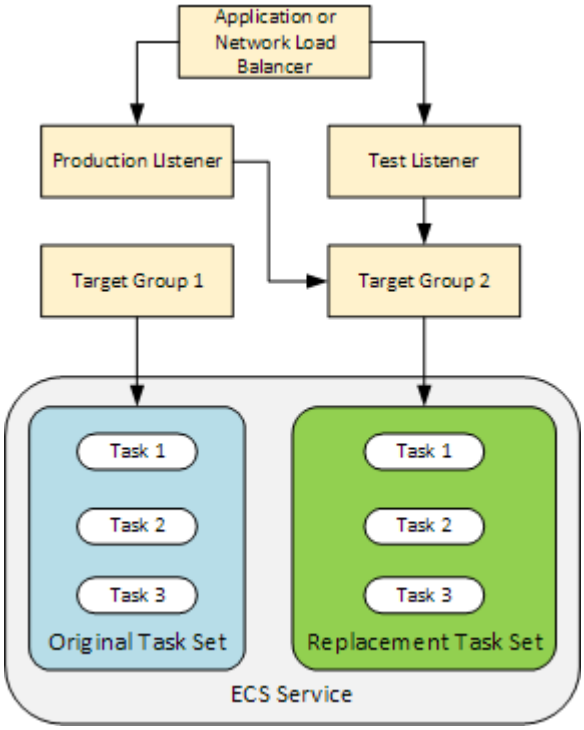
   驗證測試完成後，會發生以下其中一種情況：

   - 如果驗證失敗而轉返已設定，則會將部署狀態標示為 `Failed`，且元件在部署開始時恢復到他們的狀態。
   - 如果驗證失敗而轉返未設定，則會將部署狀態標示為 `Failed`，且元件會維持他們的目前狀態。
   - 如果驗證成功，部署會繼續至 `BeforeAllowTraffic` 勾點 。

   如需詳細資訊，請參閱使用 CloudWatch 警示監控部署 CodeDeploy、自動回復及設定部署群組的進階選項。

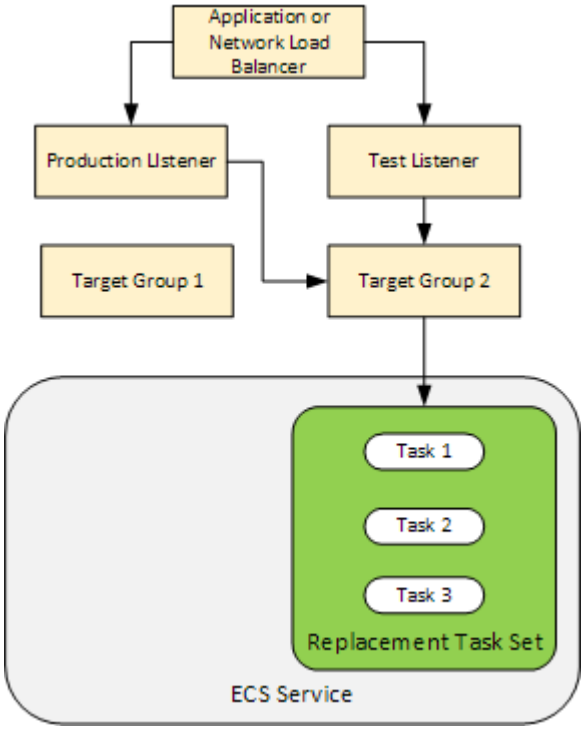
6. 執行檔案中`BeforeAllowTraffic`掛接中指定的任何 Lambda 函 AppSpec 數。

7. 已叫用 `AllowTraffic` 事件。系統會將生產流量從原始任務集重新路由到替換任務集。下圖顯示接收生產流量的替換任務集。



8. 執行檔案中`AfterAllowTraffic`掛接中指定的任何 Lambda 函 AppSpec 數。

9. 所有事件成功後，部署狀態設定為 `Succeeded`，而原始任務集則會移除。



上傳應用程式修訂

將 AppSpec 檔案放置在 Amazon S3 中，或直接將檔案輸入主控台或 AWS CLI。如需詳細資訊，請參閱 Application Specification Files。

建立應用程式和部署群組

Amazon ECS 運算平台上的 CodeDeploy 部署群組可識別接聽程式，以便為您更新的 Amazon ECS 應用程式和部署期間使用的兩個目標群組提供流量。部署群組也會定義一組組態選項，例如警示和轉返組態。

部署應用程式修訂

現在您已準備好部署部署群組中指定的更新 Amazon ECS 服務。您可以使用 CodeDeploy 控制台或創建部署命令。您可以指定參數來控制部署，包含修訂和部署群組。

更新您的申請

您可以對應用程式進行更新，然後使用 CodeDeploy 主控台或呼叫 create-deployment 命令來推送修訂。

已停止和失敗的部署

您可以使用 CodeDeploy 控制台或停止部署命令來停止部署。當您嘗試停止部署時，會發生下列三者之一：

- 部署停止，而且操作傳回成功狀態。在此情況下，不需要在已停止部署的部署群組上執行其他部署生命週期事件。
- 此部署不會立即停止，而且操作傳回擱置中狀態。在此情況下，一些部署生命週期事件可能仍然在部署群組上執行。擱置中操作完成之後，後續呼叫停止部署會傳回成功狀態。
- 部署無法停止，而且操作傳回錯誤。如需詳細資訊，請參閱 AWS CodeDeploy API 參考中的錯誤資訊和常見錯誤。

重新部署和部署復原

CodeDeploy 透過將取代工作集的流量重新路由傳送至原始工作集，以實作復原。

您可以設定部署群組以在符合特定條件時自動轉返部署 (包含部署失敗或符合警示監控閾值時)。您也可以覆寫針對個別部署中部署群組所指定的轉返設定。

您也可以手動重新部署先前部署的修訂，以選擇轉返失敗部署。

在所有情況下，新的或轉返的部署會獲指派其專屬部署 ID。主 CodeDeploy 控台會顯示自動部署結果的部署清單。

如果您重新部署，與目前部署之原始任務集相關聯的目標群組會與重新部署的替換任務集建立關聯。

如需詳細資訊，請參閱 使用以下方式重新部署和復原部署 CodeDeploy。

Amazon ECS 藍色/綠色部署 AWS CloudFormation

您可以透過管 AWS CloudFormation 理 Amazon ECS 藍色/綠色部署。 CodeDeploy如需詳細資訊，請參閱 透過以下方式建立亞馬遜ECS 藍色/綠色部署 AWS CloudFormation。

> **Note**
>
> 亞太區域 (大阪) 地區不 AWS CloudFormation 提供管理 Amazon ECS 藍/綠部署。

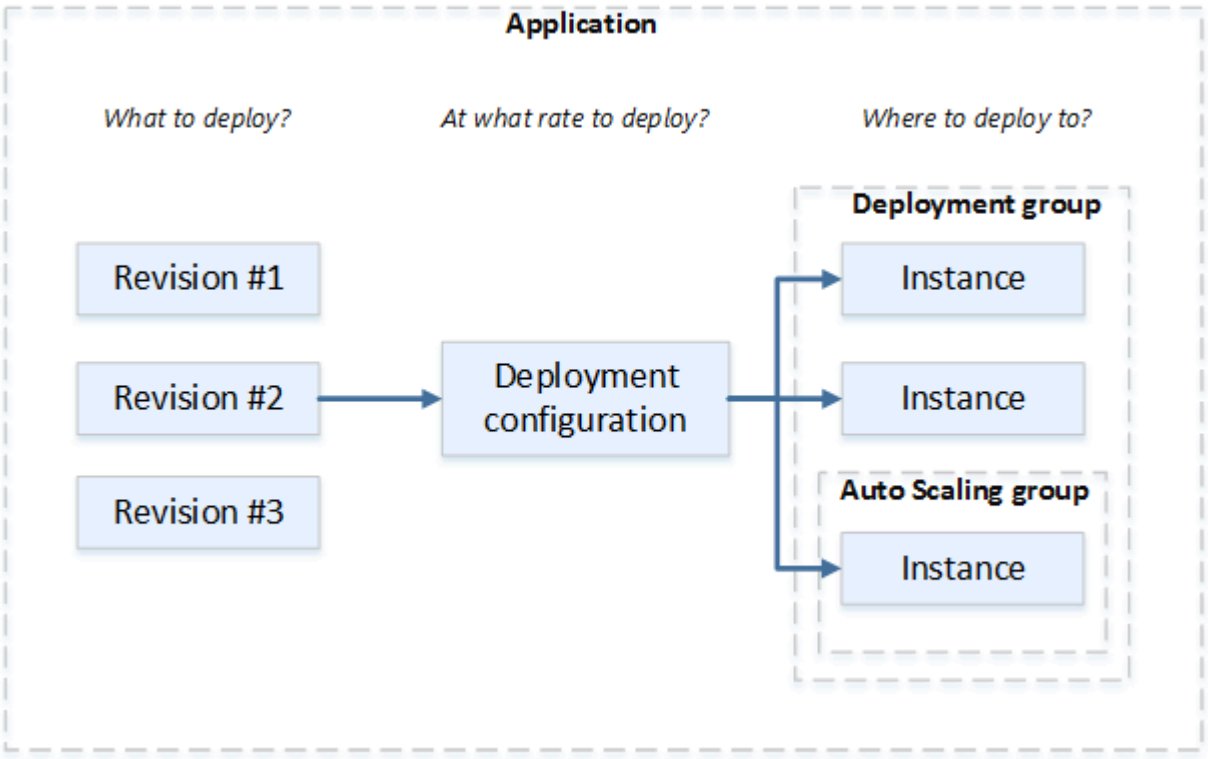
EC2/內部部署計算平台上的部署

本主題提供使用 EC2 /內部部署計算平台的 CodeDeploy 部署元件和工作流程的相關資訊。如需藍/綠部署的資訊，請參閱藍/綠部署概述。

主題

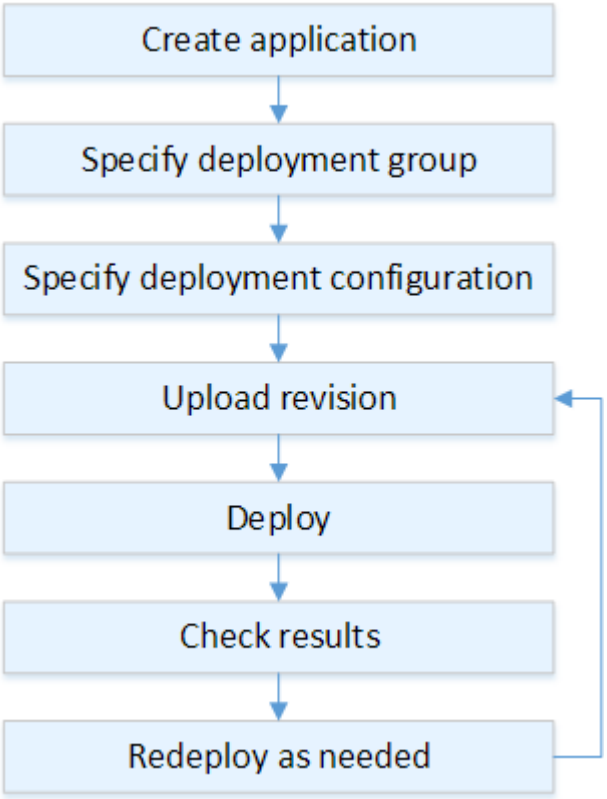
EC2/內部部署計算平台上的部署元件

下圖顯示 EC2 /內 CodeDeploy 部部署計算平台上部署的元件。



EC2/內部部署計算平台上的部署工作流程

下圖顯示應用程式修訂部署中的重要步驟：



這些步驟包括：

1. 建立應用程式，並為其指定唯一識別您要部署的應用程式修訂版本和應用程式的運算平台的名稱。CodeDeploy 在部署期間使用此名稱，以確保其參考正確的部署元件，例如部署群組、部署組態和應用程式修訂。如需詳細資訊，請參閱 建立應用程式 CodeDeploy。
2. 指定部署類型以及您要部署應用程式修訂的執行個體，來設定部署群組。就地部署會使用最新應用程式修訂來更新執行個體。藍/綠部署會使用負載平衡器來註冊部署群組的一組替換執行個體，並取消註冊原始執行個體。

   您可以指定套用至執行個體的標籤、Amazon EC2 Auto Scaling 群組名稱或兩者。

   如果您在部署群組中指定一組標記，則 CodeDeploy 會部署至少套用其中一個指定標記的執行個體。如果您指定兩個或多個標記群組，則只 CodeDeploy 會部署至符合每個標記群組準則的執行個體。如需詳細資訊，請參閱 Tagging Instances for Deployments。

   在所有情況下，執行個體都必須設定為在部署中使用 (也就是說，它們必須加上標記或屬於 Amazon EC2 Auto Scaling 群組)，並且已安裝並執行 CodeDeploy 代理程式。

   我們為您提供一個 AWS CloudFormation 範本，您可以使用這些範本快速設定基於 Amazon Linux 或 Windows 伺服器的 Amazon EC2 執行個體。我們也為您提供獨立 CodeDeploy 代理程式，以便

您可以在 Amazon Linux、Ubuntu 伺服器、RHEL (RHEL) 或 Windows 伺服器執行個體上安裝它。如需詳細資訊，請參閱 建立部署群組 CodeDeploy。

您還可以指定下列選項：

- Amazon SNS 通知。建立觸發器，在部署和執行個體中發生指定的事件 (例如成功或失敗事件) 時，傳送通知給 Amazon SNS 主題的訂閱者。如需詳細資訊，請參閱 Monitoring Deployments with Amazon SNS Event Notifications。
- 警示類型部署管理。實作 Amazon CloudWatch 警示監控，以在指標超過或低於中設定的閾值時停止部署 CloudWatch。
- 自動部署轉返。設定部署，以在部署失敗或符合警示閾值時，自動轉返先前已知良好的修訂。

3. 指定部署組態以指出應用程式修訂應同時部署至多少個執行個體，以及說明部署的成功和失敗狀況。如需詳細資訊，請參閱 View Deployment Configuration Details。
4. 將應用程式修訂版上傳到 Amazon S3 或 GitHub. 除了要部署的檔案以及在部署期間執行的任何指令碼之外，您還必須包含應用程式規格檔案 (AppSpec 檔案)。此檔案包含部署說明，例如，在何處將檔案複製至每個執行個體，以及何時執行部署指令碼。如需詳細資訊，請參閱 使用的應用程式修訂 CodeDeploy。
5. 將應用程式修訂部署至部署群組。部署群組中每個執行個體上的 CodeDeploy 代理程式會將您的應用程式修訂版從 Amazon S3 或複製 GitHub 到執行個體。然後， CodeDeploy 代理程式會解除封裝修訂，並使用該 AppSpec 檔案將檔案複製到指定的位置，並執行任何部署程序檔。如需詳細資訊，請參閱 使用建立部署 CodeDeploy。
6. 檢查部署結果。如需詳細資訊，請參閱 監控部署 CodeDeploy。
7. 重新部署修訂 如果您需要修復來源內容中的錯誤、以不同的順序執行部署指令碼，或處理失敗部署，會建議您這麼做。為此，請將修訂的來源內容、任何部署指令碼和 AppSpec 檔案重新捆綁到新的修訂版本中，然後將修訂版上傳到 Amazon S3 儲存貯體或 GitHub 儲存庫。然後將新的部署執行到具有新修訂的相同部署群組。如需詳細資訊，請參閱 使用建立部署 CodeDeploy。

設定執行個體

您必須先設定執行個體，才能第一次部署應用程式修訂。如果應用程式修訂需要三部生產伺服器和兩部備份伺服器，您要啟動或使用五個執行個體。

手動佈建執行個體：

1. 在執行個體上安裝 CodeDeploy 代理程式。 CodeDeploy 代理程式可以安裝在 Amazon Linux、Ubuntu 伺服器、RHEL 和視窗伺服器執行個體上。

2. 如果您使用標記來識別部署群組中的執行個體，請啟用標記。 CodeDeploy 依賴標記來識別執行個體並將其分組到 CodeDeploy 部署群組中。雖然入門教學使用兩者，但是您只能使用索引鍵或值來定義部署群組的標籤。
3. 使用附加 IAM 執行個體設定檔來啟動 Amazon EC2 執行個體。IAM 執行個體設定檔必須附加到 Amazon EC2 執行個體啟動時， CodeDeploy 代理程式才能驗證執行個體的身分。
4. 建立服務角色。提供服務存取權， CodeDeploy 以便展開 AWS 帳戶中的標籤。

對於初始部署， AWS CloudFormation 範本會為您完成所有這些作業。它根據 Amazon Linux 或 Windows 伺服器建立和設定新的單一 Amazon EC2 執行個體，並且已安裝 CodeDeploy 代理程式。如需詳細資訊，請參閱 使用的例證 CodeDeploy。

> **Note**
>
> 對於藍/綠部署，您可以選擇在替換環境中使用已有的執行個體，或是讓您 CodeDeploy 佈建新的執行個體做為部署程序的一部分。

上傳應用程式修訂

將 AppSpec 檔案置於應用程式的來源內容資料夾結構中的根資料夾下方。如需詳細資訊，請參閱 Application Specification Files。

將應用程式的來源內容資料夾結構綁定為封存檔案格式 (例如 zip、tar 或壓縮 tar)。將存檔檔案 (修訂版) 上傳到 Amazon S3 儲存貯體或 GitHub 儲存庫。

> **Note**
>
> 視窗伺服器執行個體不支援 tar 和壓縮的 tar 封存檔案格式 (.tar 和 .tar.gz)。

建立應用程式和部署群組

CodeDeploy 部署群組會根據執行個體的標籤、Amazon EC2 Auto Scaling 群組名稱或兩者來識別執行個體集合。多個應用程式修訂可以部署至相同的執行個體。一個應用程式修訂可以部署至多個執行個體。

例如，您可以將 "Prod" 標籤新增至三部生產伺服器，並將 "Backup" 標籤新增至兩部備份伺服器。這兩個標記可用來在應用 CodeDeploy 程式中建立兩個不同的部署群組，讓您選擇哪一組伺服器 (或兩者) 應參與部署。

您可以在部署群組中使用多個標籤群組，將部署限制為較小的一組執行個體。如需相關資訊，請參閱Tagging Instances for Deployments。

部署應用程式修訂

現在您已準備好從 Amazon S3 或 GitHub 部署群組部署應用程式修訂版。您可以使用 CodeDeploy 控制台或創建部署命令。您可以指定多個參數來控制部署 (包含修訂、部署群組和部署組態)。

更新您的申請

您可以對應用程式進行更新，然後使用 CodeDeploy 主控台或呼叫 create-deployment 命令來推送修訂。

已停止和失敗的部署

您可以使用 CodeDeploy 控制台或停止部署命令來停止部署。當您嘗試停止部署時，會發生下列三者之一：

- 部署停止，而且操作傳回成功狀態。在此情況下，不需要在已停止部署的部署群組上執行其他部署生命週期事件。一些檔案可能已複製至部署群組中的一或多個執行個體，而且可能已在其上執行一些指令碼。
- 此部署不會立即停止，而且操作傳回擱置中狀態。在此情況下，一些部署生命週期事件可能仍然在部署群組上執行。一些檔案可能已複製至部署群組中的一或多個執行個體，而且可能已在其上執行一些指令碼。擱置中操作完成之後，後續呼叫停止部署會傳回成功狀態。
- 部署無法停止，而且操作傳回錯誤。如需詳細資訊，請參閱 AWS CodeDeploy API 參考中的ErrorInformation和常見錯誤。

失敗的部署可能導致已在部署群組的一或多個執行個體上執行某些部署生命週期事件，就像已停止的部署一樣。若要瞭解部署失敗的原因，您可以使用 CodeDeploy 主控台、呼叫get-deployment-instance指令，或分析失敗部署中的記錄檔資料。如需詳細資訊，請參閱 應用程式修訂和記錄檔清理 及 檢視 CodeDeploy EC2/內部部署的記錄資料。

重新部署和部署復原

CodeDeploy 透過將先前部署的修訂版重新部署 (做為新部署) 來實作復原。

您可以設定部署群組以在符合特定條件時自動轉返部署 (包含部署失敗或符合警示監控閾值時)。您也可以覆寫針對個別部署中部署群組所指定的轉返設定。

您也可以手動重新部署先前部署的修訂，以選擇轉返失敗部署。

在所有情況下，新的或轉返的部署會獲指派其專屬部署 ID。您可以在 CodeDeploy 主控台中檢視的部署清單會顯示哪些部署是自動部署的結果。

如需更多詳細資訊，請參閱 使用以下方式重新部署和復原部署 CodeDeploy。

CodeDeploy 應用程式規格 (AppSpec) 檔案

唯一的應用程式規格AppSpec 檔案 (檔案) 是 YAML 格式或 JSON 格式的檔案。 CodeDeploy該 AppSpec 檔案用於將每個部署作為一系列生命週期事件掛接 (在檔案中定義) 來管理。

若要取得有關如何建立格式正確 AppSpec 檔案的資訊，請參閱〈〉CodeDeploy AppSpec 檔案參考。

主題

- AppSpec Amazon ECS 運算平台上的檔案
- AppSpec AWS Lambda 運算平台上的檔案
- AppSpec EC2/內部部署計算平台上的檔案
- CodeDeploy 代理程式使用 AppSpec 檔案的方式

AppSpec Amazon ECS 運算平台上的檔案

如果您的應用程式使用 Amazon ECS 運算平台，則可以使用 YAML 或 JSON 格式化 AppSpec 檔案。也可以直接輸入主控台中的編輯器。該 AppSpec 文件用於指定：

- Amazon ECS 服務的名稱，以及用來將流量導向至新任務集的容器名稱和連接埠。
- 用於驗證測試的函數。

您可以在部署生命週期事件後執行驗證 Lambda 函數。如需詳細資訊，請參閱AppSpec Amazon ECS 部署的「掛鉤」部分、 AppSpec Amazon ECS 部署的檔案結構 及 AppSpec Amazon ECS 部署的檔案範例 。

AppSpec AWS Lambda 運算平台上的檔案

如果您的應用程式使用 AWS Lambda 運算平台，則可以使用 YAML 或 JSON 格式化 AppSpec 檔案。也可以直接輸入主控台中的編輯器。該 AppSpec 文件用於指定：

- 要部署的 AWS Lambda 功能版本。
- 用於驗證測試的函數。

您可以在部署生命週期事件後執行驗證 Lambda 函數。如需詳細資訊，請參閱 AppSpec AWS Lambda 部署的「掛鉤」部分。

AppSpec EC2/內部部署計算平台上的檔案

如果您的應用程式使用 EC2 /內部部署計算平台， AppSpec 檔案一律為 YAML 格式。該 AppSpec 文件用於：

- 將應用程式修訂中的來源檔案，映射至執行個體上的目標。
- 指定已部署檔案的自訂許可。
- 指定在部署程序各階段在每個執行個體上執行的指令碼。

您可以在許多個別部署生命週期事件之後，在執行個體上執行指令碼。 CodeDeploy 只會執行檔案中指定的指令碼，但這些指令碼可以呼叫執行個體上的其他指令碼。只要執行個體上執行的作業系統支援，您就可以執行任何類型的指令碼。如需詳細資訊，請參閱 AppSpec EC2 /內部部署的「掛鉤」部分。

CodeDeploy 代理程式使用 AppSpec 檔案的方式

在部署期間， CodeDeploy 代理程式會在檔案的勾點區段中查詢目前事 AppSpec 件的名稱。如果找不到事件， CodeDeploy 代理程式會繼續進行下一個步驟。如果找到事件， CodeDeploy 代理程式會擷取要執行的指令碼清單。指令碼會按照在檔案中出現的順序依次執行。每個指令碼的狀態都會記錄在執行個體的 CodeDeploy 代理程式記錄檔中。

如果指令碼執行成功，則會傳回結束代碼 0 (零)。

> **Note**
>
> CodeDeploy 代理程式不會用於 AWS Lambda 或 Amazon ECS 部署中。

在 Install 事件期間， CodeDeploy 代理程式會使用 AppSpec 檔案檔案區段中定義的對應來決定要從修訂版複製到執行個體的資料夾或檔案。

如果作業系統上安裝的 CodeDeploy 代理程式與 AppSpec 檔案中列出的代理程式不符，則部署會失敗。

如需 CodeDeploy 代理程式記錄檔的詳細資訊，請參閱與 CodeDeploy 代理工作。

開始使用 CodeDeploy

主題

步驟 1：設定

第一次使 AWS CodeDeploy 用前，必須先完成設定步驟。這些步驟涉及建立 AWS 帳戶 (如果您還沒有帳戶)，以及具有程式設計存取權限的系統管理使用者。

在本指南中，管理使用者稱為CodeDeploy管理使用者。

註冊一個 AWS 帳戶

如果您沒有 AWS 帳戶，請完成以下步驟來建立一個。

若要註冊成為 AWS 帳戶

1. 開啟 https://portal.aws.amazon.com/billing/signup。
2. 請遵循線上指示進行。

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   當您註冊一個時 AWS 帳戶，將創建AWS 帳戶根使用者一個。根使用者有權存取該帳戶中的所有 AWS 服務 和資源。安全性最佳做法是將管理存取權指派給使用者，並僅使用 root 使用者來執行需要 root 使用者存取權的工作。

AWS 註冊過程完成後，會向您發送確認電子郵件。您可以隨時登錄 https://aws.amazon.com/ 並選擇我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

建立具有管理權限的使用者

註冊後，請保護您的 AWS 帳戶 AWS 帳戶根使用者 AWS IAM Identity Center、啟用和建立系統管理使用者，這樣您就不會將 root 使用者用於日常工作。

保護您的 AWS 帳戶根使用者

1. 選擇 Root 使用者並輸入您的 AWS 帳戶 電子郵件地址，以帳戶擁有者身分登入。AWS Management Console在下一頁中，輸入您的密碼。

   如需使用根使用者登入的說明，請參閱 AWS 登入 使用者指南中的以根使用者身分登入。

2. 若要在您的根使用者帳戶上啟用多重要素驗證 (MFA)。

   如需指示，請參閱《IAM 使用者指南》中的為 AWS 帳戶 根使用者啟用虛擬 MFA 裝置 (主控台)。

建立具有管理權限的使用者

1. 啟用 IAM Identity Center。

   如需指示，請參閱 AWS IAM Identity Center 使用者指南中的啟用 AWS IAM Identity Center。

2. 在 IAM 身分中心中，將管理存取權授予使用者。

   若要取得有關使用 IAM Identity Center 目錄 做為身分識別來源的自學課程，請參閱《使用指南》IAM Identity Center 目錄中的「以預設值設定使用AWS IAM Identity Center 者存取」。

以具有管理權限的使用者身分登入

- 若要使用您的 IAM Identity Center 使用者簽署，請使用建立 IAM Identity Center 使用者時傳送至您電子郵件地址的簽署 URL。

  如需使用 IAM 身分中心使用者登入的說明，請參閱使用AWS 登入 者指南中的登入 AWS 存取入口網站。

指派存取權給其他使用者

1. 在 IAM 身分中心中，建立遵循套用最低權限許可的最佳做法的權限集。

   如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「建立權限集」。

2. 將使用者指派給群組，然後將單一登入存取權指派給群組。

   如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「新增群組」。

您現在已建立並以CodeDeploy 系統管理使用者身分登入。

授與程式設計存取權

如果使用者想要與 AWS 之外的 AWS Management Console. 授與程式設計存 AWS取權的方式取決於正在存取的使用者類型。

若要授與使用者程式設計存取權，請選擇下列其中一個選項。

哪個使用者需要程式設計存取權？	到	By
人力身分 (IAM Identity Center 中管理的使用者)	使用臨時登入資料來簽署對 AWS CLI、 AWS SDK 或 AWS API 的程式設計要求。	請依照您要使用的介面所提供的指示操作。 • 如需詳細資訊 AWS CLI，請參閱《使 AWS CLI 用AWS Command Line Interface者指南》 AWS IAM Identity Center中的〈配置使用〉。 • 如需 AWS SDK、工具和 AWS API，請參閱 AWS SDK 和工具參考指南中的 IAM 身分中心身分驗證。
IAM	使用臨時登入資料來簽署對 AWS CLI、 AWS SDK 或 AWS API 的程式設計要求。	遵循《IAM 使用者指南》中的〈將臨時登入資料搭配 AWS 資源使用〉中的指示
IAM	(不建議使用) 使用長期認證來簽署對 AWS CLI、 AWS SDK 或 AWS API 的程式設計要求。	請依照您要使用的介面所提供的指示操作。 • 如需相關資訊 AWS CLI，請參閱使用指南中的使用 IAM 使用者登入資料進行驗證。AWS Command Line Interface • 對於 AWS SDK 和工具，請參閱 AWS SDK 和工具參

哪個使用者需要程式設計存取權？	到	By
		考指南中的使用長期憑據進行身份驗證。 • 如需 AWS API，請參閱 IAM 使用者指南中的管理 IAM 使用者的存取金鑰。

> **Important**
>
> 強烈建議您將 CodeDeploy 管理員使用者設定為員工身分 (在 IAM 身分中心管理的使用者)。AWS CLI本指南中的許多程序假設您正在使用 AWS CLI 來執行配置。

> **Important**
>
> 如果您設定 AWS CLI，系統可能會提示您指定「 AWS 區域」。選擇「區域」中列出的其中一個支援區域以及中的端點AWS 一般參考。

步驟 2：建立服務角色 CodeDeploy

在中 AWS，服務角色可用來授與 AWS 服務的權限，以便它可以存取 AWS 資源。您附加至服務角色的原則會決定服務可存取哪些資源，以及它可以使用這些資源的作業。

您為其建立的服務角色 CodeDeploy 必須獲得運算平台所需的權限。如果您部署到多個計算平台，請為每個計算平台建立一個服務角色。若要新增權限，請附加下列 AWS 提供的一或多個原則：

對於 EC2/ 內部部署，請附加原則。**AWSCodeDeployRole**此政策提供您服務角色執行下列作業的許可：

- 閱讀執行個體上的標籤，或透過 Amazon EC2 Auto Scaling 群組名稱識別您的 Amazon EC2 執行個體。
- 讀取、建立、更新和刪除 Amazon EC2 Auto Scaling 群組、生命週期勾點和擴展政策。
- 將資訊發佈到 Amazon SNS 主題。
- 擷取有關 CloudWatch 警報的資訊。

- 讀取並更新 Elastic Load Balancing。

> **Note**
>
> 如果您使用啟動範本建立 Auto Scaling 群組，則必須新增下列權限：
>
> - `ec2:RunInstances`
> - `ec2:CreateTags`
> - `iam:PassRole`
>
> 如需詳細資訊步驟 2：建立服務角色，請參閱 Amazon EC2 Auto Scaling 使用者指南中的為自動擴展群組建立啟動範本和啟動範本支援。

對於 Amazon ECS 部署，如果您想要完整存取支援服務，請附加**AWSCodeDeployRoleForECS**政策。此政策提供您服務角色執行下列作業的許可：

- 讀取、更新和刪除 Amazon ECS 任務集。
- 更新 Elastic Load Balancing 目標群組、監聽器和規則。
- 調用 AWS Lambda 函數。
- 存取 Amazon S3 儲存貯體中的修訂版檔案。
- 擷取有關 CloudWatch 警報的資訊。
- 將資訊發佈到 Amazon SNS 主題。

對於 Amazon ECS 部署，如果您想要有限的支援服務存取權，請附加**AWSCodeDeployRoleForECSLimited**政策。此政策提供您服務角色執行下列作業的許可：

- 讀取、更新和刪除 Amazon ECS 任務集。
- 擷取有關 CloudWatch 警報的資訊。
- 將資訊發佈到 Amazon SNS 主題。

對於 AWS Lambda 部署，如果您想要允許發佈到 Amazon SNS，請附加**AWSCodeDeployRoleForLambda**政策。此政策提供您服務角色執行下列作業的許可：

- 讀取，更新和調用 AWS Lambda 函數和別名。
- 存取 Amazon S3 儲存貯體中的修訂版檔案。
- 擷取有關 CloudWatch 警報的資訊。

- 將資訊發佈到 Amazon SNS 主題。

對於 AWS Lambda 部署，如果您想限制對 Amazon SNS 的存取，請附加**AWSCodeDeployRoleForLambdaLimited**政策。此政策提供您服務角色執行下列作業的許可：

- 讀取，更新和調用 AWS Lambda 函數和別名。
- 存取 Amazon S3 儲存貯體中的修訂版檔案。
- 擷取有關 CloudWatch 警報的資訊。

做為設定服務角色的一部分，您也需要更新其信任關係，指定您希望授予其存取的端點。

您可以使用 IAM 主控台 AWS CLI、或 IAM API 建立服務角色。

主題

建立服務角色 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。
3. 選擇AWS 服務，然後在使用案例下，從下拉式清單中選擇CodeDeploy。
4. 選擇您的使用案例：
   - 對於 EC2/內部部署，請選擇。CodeDeploy
   - 若為 AWS Lambda 部署，請選擇CodeDeploy 使用 Lambda。
   - 對於 Amazon ECS 部署，請選擇 CodeDeploy -EC S。
5. 選擇下一步。
6. 在 [新增權限] 頁面上，會顯示使用案例的正確權限原則。選擇下一步。
7. 在 [名稱、檢閱和建立] 頁面的 [角色名稱] 中，輸入服務角色的名稱 (例如，**CodeDeployServiceRole**)，然後選擇 [建立角色]。

您也可以在角色說明中輸入此服務角色的說明。

8. 若您希望此服務角色擁有存取目前所有支援端點的許可，您即已完成此程序。

   若要限制此服務角色無法存取某些端點，請繼續執行此程序中的其餘步驟。

9. 在角色清單中，搜尋並選擇您剛建立的角色 (CodeDeployServiceRole)。
10. 選擇信任關係標籤。
11. 選擇編輯信任政策。

    您應該會看到以下政策，提供服務角色所有支援端點的存取許可：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "codedeploy.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

若要僅將服務角色存取權授與某些受支援的端點，請將信任策略文字方塊的內容取代為下列策略。移除您要阻止存取的端點線，然後選擇 [更新策略]。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": [

                    "codedeploy.us-east-1.amazonaws.com",
                    "codedeploy.us-east-2.amazonaws.com",
```

```
                    "codedeploy.us-west-1.amazonaws.com",
                    "codedeploy.us-west-2.amazonaws.com",
                    "codedeploy.ca-central-1.amazonaws.com",
                    "codedeploy.ap-east-1.amazonaws.com",
                    "codedeploy.ap-northeast-1.amazonaws.com",
                    "codedeploy.ap-northeast-2.amazonaws.com",
                    "codedeploy.ap-northeast-3.amazonaws.com",
                    "codedeploy.ap-southeast-1.amazonaws.com",
                    "codedeploy.ap-southeast-2.amazonaws.com",
                    "codedeploy.ap-southeast-3.amazonaws.com",
                    "codedeploy.ap-southeast-4.amazonaws.com",
                    "codedeploy.ap-south-1.amazonaws.com",
                    "codedeploy.ap-south-2.amazonaws.com",
                    "codedeploy.ca-central-1.amazonaws.com",
                    "codedeploy.eu-west-1.amazonaws.com",
                    "codedeploy.eu-west-2.amazonaws.com",
                    "codedeploy.eu-west-3.amazonaws.com",
                    "codedeploy.eu-central-1.amazonaws.com",
                    "codedeploy.eu-central-2.amazonaws.com",
                    "codedeploy.eu-north-1.amazonaws.com",
                    "codedeploy.eu-south-1.amazonaws.com",
                    "codedeploy.eu-south-2.amazonaws.com",
                    "codedeploy.il-central-1.amazonaws.com",
                    "codedeploy.me-central-1.amazonaws.com",
                    "codedeploy.me-south-1.amazonaws.com",
                    "codedeploy.sa-east-1.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

如需有關建立服務角色的詳細資訊，請參閱《IAM 使用者指南》中的建立角色以將權限委派給 AWS 服務。

建立服務角色 (CLI)

1. 在您的開發機器上，建立 (舉例) 名為 CodeDeployDemo-Trust.json 的文字檔案。此文件用於允 CodeDeploy 許代表您工作。

   執行以下任意一項：

- 若要授與存取所有支援的 AWS 區域，請將下列內容儲存在檔案中：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "codedeploy.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

- 若要僅授予存取某些支援的區域，請在檔案中輸入以下內容，並移除您希望排除存取的區域行：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": [

                    "codedeploy.us-east-1.amazonaws.com",
                    "codedeploy.us-east-2.amazonaws.com",
                    "codedeploy.us-west-1.amazonaws.com",
                    "codedeploy.us-west-2.amazonaws.com",
                    "codedeploy.ca-central-1.amazonaws.com",
                    "codedeploy.ap-east-1.amazonaws.com",
                    "codedeploy.ap-northeast-1.amazonaws.com",
                    "codedeploy.ap-northeast-2.amazonaws.com",
                    "codedeploy.ap-northeast-3.amazonaws.com",
                    "codedeploy.ap-southeast-1.amazonaws.com",
                    "codedeploy.ap-southeast-2.amazonaws.com",
                    "codedeploy.ap-southeast-3.amazonaws.com",
                    "codedeploy.ap-southeast-4.amazonaws.com",
```

```
                    "codedeploy.ap-south-1.amazonaws.com",
                    "codedeploy.ap-south-2.amazonaws.com",
                    "codedeploy.ca-central-1.amazonaws.com",
                    "codedeploy.eu-west-1.amazonaws.com",
                    "codedeploy.eu-west-2.amazonaws.com",
                    "codedeploy.eu-west-3.amazonaws.com",
                    "codedeploy.eu-central-1.amazonaws.com",
                    "codedeploy.eu-central-2.amazonaws.com",
                    "codedeploy.eu-north-1.amazonaws.com",
                    "codedeploy.eu-south-1.amazonaws.com",
                    "codedeploy.eu-south-2.amazonaws.com",
                    "codedeploy.il-central-1.amazonaws.com",
                    "codedeploy.me-central-1.amazonaws.com",
                    "codedeploy.me-south-1.amazonaws.com",
                    "codedeploy.sa-east-1.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

> **Note**
>
> 請不要在清單中的最後一個端點之後使用逗點。

2. 從相同目錄裡，呼叫 create-role 命令，根據您剛才建立的文字檔案資訊，建立名為 **`CodeDeployServiceRole`** 的服務角色：

```
aws iam create-role --role-name CodeDeployServiceRole --assume-role-policy-document
 file://CodeDeployDemo-Trust.json
```

> **Important**
>
> 請確認在檔案名稱之前包含 `file://`。這是此命令必要項目。

在命令的輸出中，記下 `Role` 物件下 `Arn` 項目的值。您稍後將需要它來建立部署群組。若您忘記該值，請遵循取得服務角色 ARN (CLI) 中的說明。

3. 您使用的受管原則取決於運算平台。

- 如果您的部署是 EC2 /內部部署計算平台：

  呼叫命attach-role-policy令，根據名為**CodeDeployServiceRole**的 IAM 受管政策授予名為許可的服務角色**AWSCodeDeployRole**。例如：

  ```
  aws iam attach-role-policy --role-name CodeDeployServiceRole --policy-arn
   arn:aws:iam::aws:policy/service-role/AWSCodeDeployRole
  ```

- 如果您的部署是使用 AWS Lambda 運算平台：

  呼叫命attach-role-policy令，根據名為**CodeDeployServiceRoleAWSCodeDeployRoleForLambda**或的 IAM 受管政策提供名為許可的服務角色**AWSCodeDeployRoleForLambdaLimited**。例如：

  ```
  aws iam attach-role-policy --role-name CodeDeployServiceRole --policy-arn
   arn:aws:iam::aws:policy/service-role/AWSCodeDeployRoleForLambda
  ```

- 如果您的部署是 Amazon ECS 運算平台：

  呼叫命attach-role-policy令，根據名為**CodeDeployServiceRoleAWSCodeDeployRoleForECS**或的 IAM 受管政策提供名為許可的服務角色**AWSCodeDeployRoleForECSLimited**。例如：

  ```
  aws iam attach-role-policy --role-name CodeDeployServiceRole --policy-arn
   arn:aws:iam::aws:policy/AWSCodeDeployRoleForECS
  ```

如需建立服務角色的詳細資訊，請參閱《IAM 使用者指南》中的〈建立 AWS 服務角色〉。

取得服務角色 ARN (主控台)

若要使用 IAM 主控台取得服務角色的 ARN：

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇角色。
3. 在 Filter (篩選條件) 方塊中，輸入 **CodeDeployServiceRole**，然後按 Enter 鍵。
4. 選擇CodeDeployServiceRole。
5. 記下 Role ARN (角色 ARN) 欄位的值。

取得服務角色 ARN (CLI)

若要使用取得服務角色的 ARN，請針對名**CodeDeployServiceRole**為 AWS CLI 以下服務角色呼叫get-role命令：

```
aws iam get-role --role-name CodeDeployServiceRole --query "Role.Arn" --output text
```

傳回的值即為服務角色的 ARN。

步驟 3：限制 CodeDeploy 使用者的權限

基於安全理由，建議您將在中建立的管理使用者的權限限制為僅限步驟 1：設定於在中建立和管理部署所需的權限 CodeDeploy。

請使用下列一系列程序來限制系 CodeDeploy 統管理使用者的權限。

開始之前

- 請確定您已按照中的指示在 IAM 身分中心建立 CodeDeploy 管理使用者步驟 1：設定。

建立許可集合

您稍後會將此權限集指派給系 CodeDeploy 統管理使用者。

1. 請登入 AWS Management Console 並開啟 AWS IAM Identity Center 主控台，網址為 https://console.aws.amazon.com/singlesignon/。
2. 在瀏覽窗格中，選擇 [權限集]，然後選擇 [建立權限集]。
3. 選擇 [自訂權限集]。
4. 選擇下一步。
5. 選擇內嵌政策。
6. 移除範例程式碼。
7. 新增下列原則程式碼：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CodeDeployAccessPolicy",
```

```
      "Effect": "Allow",
      "Action": [
        "autoscaling:*",
        "codedeploy:*",
        "ec2:*",
        "lambda:*",
        "ecs:*",
        "elasticloadbalancing:*",
        "iam:AddRoleToInstanceProfile",
        "iam:AttachRolePolicy",
        "iam:CreateInstanceProfile",
        "iam:CreateRole",
        "iam:DeleteInstanceProfile",
        "iam:DeleteRole",
        "iam:DeleteRolePolicy",
        "iam:GetInstanceProfile",
        "iam:GetRole",
        "iam:GetRolePolicy",
        "iam:ListInstanceProfilesForRole",
        "iam:ListRolePolicies",
        "iam:ListRoles",
        "iam:PutRolePolicy",
        "iam:RemoveRoleFromInstanceProfile",
        "s3:*",
        "ssm:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CodeDeployRolePolicy",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::account-ID:role/CodeDeployServiceRole"
    }
  ]
}
```

在此原則中，請將 *arn: aw: iam:: ## ID: ##/############ CodeDeployServiceRole*的 ARN 值。 CodeDeploy 步驟 2：建立服務角色 CodeDeploy您可以在 IAM 主控台中服務角色的詳細資料頁面中找到 ARN 值。

上述政策可讓您將應用程式部署到 AWS Lambda 運算平台、EC2 /內部部署運算平台和 Amazon ECS 運算平台。

您可以使用本文件中提供的 AWS CloudFormation 範本來啟動相容的 Amazon EC2 執行個體 CodeDeploy。若要使用 AWS CloudFormation 範本來建立應用程式、部署群組或部署設定，您必須將`cloudformation:*`權限新增至 AWS CloudFormation CodeDeploy 系統管理使用者的權限原則，以及提供相 AWS CloudFormation 依 AWS 服務和動作的存取權，如下所示：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        ...
        "cloudformation:*"
      ],
      "Resource": "*"
    }
  ]
}
```

8. 選擇下一步。
9. 在權限集名稱中，輸入：

```
CodeDeployUserPermissionSet
```

10. 選擇下一步。
11. 在 [檢閱並建立] 頁面上，檢閱資訊並選擇 [建立]。

將權限集指派給 CodeDeploy 系統管理使用者

1. 在功 AWS 帳戶 能窗格中，選擇 AWS 帳戶，然後選取您目前登入的對象旁邊的核取方塊。
2. 選擇「指派使用者或群組」按鈕。
3. 選擇 Users (使用者) 索引標籤。
4. 選取 CodeDeploy 管理使用者旁邊的核取方塊。
5. 選擇下一步。
6. 選取旁邊的核取方塊`CodeDeployUserPermissionSet`。

7. 選擇下一步。
8. 複查資訊，然後選擇「提交」。

   現在，您已經分配了 CodeDeploy 管理用戶和您CodeDeployUserPermissionSet的 AWS 帳戶，將它們綁定在一起。

若要以 CodeDeploy 系統管理使用者的身分登出並重新登入

1. 登出之前，請確定您擁有 CodeDeploy 管理員使用者的 AWS 存取入口網站 URL 以及使用者名稱和一次性密碼。

   > **Note**
   >
   > 如果您沒有這些信息，請轉到 IAM 身份中心的 CodeDeploy 管理用戶詳細信息頁面，選擇重置密碼，生成一次性密碼 [...] ，然後再次重設密碼以在螢幕上顯示資訊。

2. 登出 AWS。
3. 將 AWS 訪問門戶網址粘貼到瀏覽器的地址欄中。
4. 以管理員使用者身 CodeDeploy 分登入。

   螢幕上會出現一個AWS 帳戶方塊。
5. 選擇 AWS 帳戶，然後選擇您指派 CodeDeploy 管 AWS 帳戶 理員使用者和權限集的名稱。
6. 在旁邊CodeDeployUserPermissionSet，選擇 [管理主控台]。

   AWS Management Console 隨即出現。您現在已以有 CodeDeploy 限權限的管理使用者身分登入。您現在可以以此使用者身分執行 CodeDeploy相 CodeDeploy關作業，而且只能執行相關作業。

步驟 4：為您的 Amazon EC2 執行個體建立 IAM 執行個體設定檔

> **Note**
>
> 如果您使用的是 Amazon ECS 或 AWS Lambda 運算平台，請跳過此步驟。

您的 Amazon EC2 執行個體需要許可才能存取存 GitHub 放應用程式的 Amazon S3 儲存貯體或儲存庫。若要啟動與相容的 Amazon EC2 執行個體 CodeDeploy，您必須建立其他 IAM 角色 (執行個體

設定檔)。這些指示說明如何建立 IAM 執行個體設定檔以連接到 Amazon EC2 執行個體。此角色授予 CodeDeploy 代理程式存取存 GitHub 放應用程式之 Amazon S3 儲存貯體或儲存庫的權限。

您可以使用 IAM 主控台或 IAM API 建立 IAM 執行個體設定檔。 AWS CLI

> **Note**
>
> 您可以將 IAM 執行個體設定檔連接至啟動的 Amazon EC2 執行個體或先前啟動的執行個體。有關詳情，請參閱執行個體設定檔。

主題

為您的 Amazon EC2 執行個體 (CLI) 建立 IAM 執行個體設定檔

在這些步驟中，假設您已經遵循開始使用 CodeDeploy 中前三個步驟的說明。

1. 在您的開發機器上，建立名為 `CodeDeployDemo-EC2-Trust.json` 的文字檔案。貼上下列內容，讓 Amazon EC2 代表您運作：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

2. 在相同的目錄中，建立名為 `CodeDeployDemo-EC2-Permissions.json` 的文字檔案。貼上下列內容：

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

Note

我們建議您將此政策限制為只有 Amazon EC2 執行個體必須存取的 Amazon S3 儲存貯體。確保授予對包含 CodeDeploy 代理程式的 Amazon S3 儲存貯體的存取權。否則，在執行個體上安裝或更新 CodeDeploy 代理程式時，可能會發生錯誤。若要僅授與 IAM 執行個體設定檔存取權限給 Amazon S3 中的某些 CodeDeploy 資源套件儲存貯體，請使用下列政策，但移除要阻止存取的儲存貯體的行：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3:::replace-with-your-s3-bucket-name/*",
        "arn:aws:s3:::aws-codedeploy-us-east-2/*",
        "arn:aws:s3:::aws-codedeploy-us-east-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-2/*",
        "arn:aws:s3:::aws-codedeploy-ca-central-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-3/*",
        "arn:aws:s3:::aws-codedeploy-eu-central-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-central-2/*",
```

```
        "arn:aws:s3:::aws-codedeploy-eu-north-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-south-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-south-2/*",
        "arn:aws:s3:::aws-codedeploy-il-central-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-east-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-3/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-3/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-4/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-2/*",
        "arn:aws:s3:::aws-codedeploy-me-central-1/*",
        "arn:aws:s3:::aws-codedeploy-me-south-1/*",
        "arn:aws:s3:::aws-codedeploy-sa-east-1/*"
      ]
    }
  ]
}
```

> **Note**
>
> 如果您想要使用 IAM 授權或 Amazon Virtual Private Cloud (VPC) 端點搭配使用 CodeDeploy，則需要新增更多許可。如需詳細資訊，請參閱 CodeDeploy 搭配 Amazon Virtual Private Cloud 使用。

3. 從相同的目錄中，呼叫create-role命令，根據第一個檔案中的資訊建立名**`CodeDeployDemo-EC2-Instance-Profile`**為的 IAM 角色：

> **Important**
>
> 請確認在檔案名稱之前包含 `file://`。這是此命令必要項目。

```
aws iam create-role --role-name CodeDeployDemo-EC2-Instance-Profile --assume-role-policy-document file://CodeDeployDemo-EC2-Trust.json
```

4. 從相同的目錄中，呼叫 put-role-policy 命令，根據第二個檔案中的資訊，將許可提供給名為 **CodeDeployDemo-EC2-Instance-Profile** 的角色：

   > **Important**
   >
   > 請確認在檔案名稱之前包含 `file://`。這是此命令必要項目。

   ```
   aws iam put-role-policy --role-name CodeDeployDemo-EC2-Instance-Profile --policy-
   name CodeDeployDemo-EC2-Permissions --policy-document file://CodeDeployDemo-EC2-
   Permissions.json
   ```

5. 呼叫以授attach-role-policy予角色 Amazon EC2 Systems Manager 權限，以便 SSM 可以安裝 CodeDeploy 代理程式。如果您計劃使用命令列從公有 Amazon S3 儲存貯體安裝代理程式，則不需要此政策。進一步了解安裝 CodeDeploy 代理程式。

   ```
   aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
   AmazonSSMManagedInstanceCore --role-name CodeDeployDemo-EC2-Instance-Profile
   ```

6. 呼叫命create-instance-profile令後跟指add-role-to-instance-profile令來建立名為的 IAM 執行個體設定檔**CodeDeployDemo-EC2-Instance-Profile**。執行個體設定檔允許 Amazon EC2 在執行個體首次啟動時，**CodeDeployDemo-EC2-Instance-Profile**將名為的 IAM 角色傳遞給 Amazon EC2 執行個體：

   ```
   aws iam create-instance-profile --instance-profile-name CodeDeployDemo-EC2-
   Instance-Profile
   aws iam add-role-to-instance-profile --instance-profile-name CodeDeployDemo-EC2-
   Instance-Profile --role-name CodeDeployDemo-EC2-Instance-Profile
   ```

   如果您需要取得 IAM 執行個體設定檔的名稱，請參閱AWS CLI 參考資料的 IAM 一節中的 list-instance-profiles-for-role。

您現在已建立 IAM 執行個體設定檔以連接到 Amazon EC2 執行個體。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的適用於 Amazon EC2 的 IAM 角色。

為您的 Amazon EC2 執行個體 (主控台) 建立 IAM 執行個體設定檔

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在 IAM 主控台的導覽窗格中，選擇 [政策]，然後選擇 [建立政策]。
3. 在指定許可頁面上，選擇 JSON。
4. 移除範例JSON程式碼。
5. 貼上以下程式碼：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 我們建議您將此政策限制為只有 Amazon EC2 執行個體必須存取的 Amazon S3 儲存貯體。確保授予對包含 CodeDeploy 代理程式的 Amazon S3 儲存貯體的存取權。否則，在執行個體上安裝或更新 CodeDeploy 代理程式時，可能會發生錯誤。若要僅授與 IAM 執行個體設定檔存取權限給 Amazon S3 中的某些 CodeDeploy 資源套件儲存貯體，請使用下列政策，但移除要阻止存取的儲存貯體的行：
>
> ```
> {
>   "Version": "2012-10-17",
>   "Statement": [
>     {
>       "Effect": "Allow",
>       "Action": [
>         "s3:Get*",
>         "s3:List*"
> ```

```
      ],
      "Resource": [
        "arn:aws:s3:::replace-with-your-s3-bucket-name/*",
        "arn:aws:s3:::aws-codedeploy-us-east-2/*",
        "arn:aws:s3:::aws-codedeploy-us-east-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-2/*",
        "arn:aws:s3:::aws-codedeploy-ca-central-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-3/*",
        "arn:aws:s3:::aws-codedeploy-eu-central-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-central-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-north-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-south-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-south-2/*",
        "arn:aws:s3:::aws-codedeploy-il-central-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-east-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-3/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-3/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-4/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-2/*",
        "arn:aws:s3:::aws-codedeploy-me-central-1/*",
        "arn:aws:s3:::aws-codedeploy-me-south-1/*",
        "arn:aws:s3:::aws-codedeploy-sa-east-1/*"
      ]
    }
  ]
}
```

> **Note**
>
> 如果您想要使用 IAM 授權或 Amazon Virtual Private Cloud (VPC) 端點搭配使用 CodeDeploy，則需要新增更多許可。如需詳細資訊，請參閱 CodeDeploy 搭配 Amazon Virtual Private Cloud 使用。

6. 選擇下一步。
7. 在 [檢閱並建立] 頁面的 [原則名稱] 方塊中，輸入**CodeDeployDemo-EC2-Permissions**。
8. (選用) 針對 Description (描述)，輸入政策的描述。
9. 選擇 Create policy (建立政策)。
10. 在導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。
11. 在使用案例下，選擇 EC2 使用案例。
12. 選擇下一步。
13. 在原則清單中，選取您剛建立之原則旁邊的核取方塊 (CodeDeployDemo-EC2 權限)。如有需要，請使用搜尋方塊來尋找政策。
14. 若要使用 Systems Manager 來安裝或設定 CodeDeploy 代理程式，請選取 Amazon ManagedInstanceCore SSM 旁邊的核取方塊。此 AWS 受管原則可讓執行個體使用 Systems Manager 服務核心功能。如有需要，請使用搜尋方塊來尋找政策。如果您計劃使用命令列從公有 Amazon S3 儲存貯體安裝代理程式，則不需要此政策。進一步了解安裝 CodeDeploy 代理程式。
15. 選擇下一步。
16. 在 [名稱、檢閱和建立] 頁面的 [角色名稱] 中，輸入服務角色的名稱 (例如，**CodeDeployDemo-EC2-Instance-Profile**)，然後選擇 [建立角色]。

    您也可以在角色說明中輸入此服務角色的說明。

您現在已建立 IAM 執行個體設定檔以連接到 Amazon EC2 執行個體。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的適用於 Amazon EC2 的 IAM 角色。

產品與服務整合 CodeDeploy

根據預設，會 CodeDeploy 與許多 AWS 服務和合作夥伴產品和服務整合。下列資訊可協助您設定 CodeDeploy 為與您使用的產品和服務整合。

與其他 AWS 服務整合

CodeDeploy 與以下 AWS 服務集成：

Amazon CloudWatch	Amazon CloudWatch 是 AWS 雲端資源和執行應用程式的監控服務 AWS。您可以使用 Amazon CloudWatch 收集和追蹤指標、收集和監控日誌檔，以及設定警示。 CodeDeploy 支援下列 CloudWatch 工具： • CloudWatch 當您指定的監視指標超過或低於您在警示規則中指定的閾值時，監視部署並停止它們的 CloudWatch 警示。若要使用警示監控，請先在中設定警示 CloudWatch，然後將其新增 CodeDeploy 至應用程式或部署群組，在該應用程式或部署群組中，在該群組中應在警示啟動時停止。 進一步了解： • 建立 CloudWatch 記錄檔警示 • Amazon E CloudWatch vents 可偵測執行個體狀態或 CodeDeploy 作業中部署的變更，並對其做出反應。然後，根據您建立的規則，當部署或執行個體進入您在規則中指定的狀態時，E CloudWatch vents 會叫用一或多個目標動作。

	進一步了解： • 使用 Amazon CloudWatch 事件監控部署 • Amazon CloudWatch 日誌可監控 CodeDeploy 代理程式建立的三種類型日誌，而無需一次登入一個執行個體。 進一步了解： • 在 CodeDeploy 記錄主控台中檢視 CloudWatch 記錄
Amazon EC2 Auto Scaling	CodeDeploy 支援 Amazon EC2 Auto Scaling。此 AWS 服務可以根據您指定的條件自動啟動 Amazon EC2 執行個體，例如： • 超過指定 CPU 使用率的限制。 • 磁碟讀取或寫入。 • 傳入或傳出的網路流量超過指定時間間隔。 您可以在需要時向外擴展一組 Amazon EC2 執行個體，然後用它們自動將應用程式修訂部署 CodeDeploy 到這些執行個體。當這些 Amazon EC2 執行個體不再需要時，Amazon EC2 Auto Scaling 會終止這些執行個體。 進一步了解： • CodeDeploy 與 Amazon EC2 Auto Scaling 集成 • 教學課程：用 CodeDeploy 於將應用程式部署到 Auto Scaling 群組 • 引擎蓋下： CodeDeploy 和 Auto Scaling 集成

Amazon Elastic Container Service	您可以使用 CodeDeploy 將 Amazon ECS 容器化應用程式部署為任務集。 CodeDeploy 將應用程式的更新版本安裝為新的取代工作集，以執行藍/綠部署。 CodeDeploy 將原始應用程式工作集的生產流量重新路由傳送至取代工作集。成功部署後，原始任務集會終止。有關 Amazon ECS 的更多信息，請參閱 Amazon 彈性容器服務。 您可以選擇初期測試、線性或規劃，來管理部署期間流量轉移至更新工作集的方 all-at-once 式。如需 Amazon ECS 部署的詳細資訊，請參閱 Amazon ECS 運算平台上的部署。
AWS CloudTrail	CodeDeploy 與整合AWS CloudTrail。此服務會擷取您帳戶中由或代表您 AWS 帳戶發出 CodeDeploy 的 API 呼叫，並將日誌檔傳送到您指定的 Amazon S3 儲存貯體。 CloudTrail 從 CodeDeploy 主控台、透過 CodeDeploy 命令擷取 API 呼叫 AWS CLI，或直接從 API 擷取 CodeDeploy API 呼叫。使用收集的資訊 CloudTrail，您可以判斷： • 提出了哪個要求 CodeDeploy。 • 提出請求的來源 IP 地址。 • 提出要求的人員。 • 提出時間。 進一步了解： • Monitoring Deployments

AWS Cloud9	AWS Cloud9是一種線上、雲端整合式開發環境 (IDE)，您只需使用連線網際網路的電腦上的瀏覽器即可撰寫、執行、偵錯和部署程式碼。AWS Cloud9 包含程式碼編輯器、偵錯工具、終端機和基本工具，例如 AWS CLI 和 Git。 • 您可以使用 AWS Cloud9 IDE 執行、偵錯和建置 GitHub 儲存庫中的程式碼。您可以檢視、變更和儲存程式碼，方法是使用其 IDE Environment (環境) 視窗及編輯器標籤。準備就緒後，您可以在 AWS Cloud9 終端機工作階段中使用 Git，將程式碼變更推送至 GitHub 儲存庫，然後用 AWS CodeDeploy 來部署更新。如需 AWS Cloud9 與搭配使用的詳細資訊 GitHub，請參閱的GitHub 範例 AWS Cloud9。 • 您可以使用 AWS Cloud9 IDE 來更新 AWS Lambda 函數。然後，您可以使用 AWS CodeDeploy 來建立將流量轉移到新版本 AWS Lambda 函數的部署。如需詳細資訊，請參閱在 AWS Cloud9 整合式開發環境 (IDE) 中使用 AWS Lambda 函數。 如需有關的詳細資訊 AWS Cloud9，請參閱「什麼是」 AWS Cloud9和「入門使用」 AWS Cloud9。

AWS CodePipeline	AWS CodePipeline 是一種持續交付的服務，讓您能夠將發行軟體所需的步驟，依持續交付程序進行模型化、視覺化和自動化。您可以使用 AWS CodePipeline 定義您自己的發佈程序，讓服務能夠在每次程式碼變更時，建置、測試與部署您的代碼。例如，您可能有三個應用程式適用的部署群組：Beta、Gamma 和 Prod。您可以設定管道，讓每次原始碼發生變更時，一個一個地將更新部署到每個部署群組。 您可以配置 AWS CodePipeline 為用 CodeDeploy 於部署： • 程式碼傳送至 Amazon EC2 執行個體、現場部署執行個體或兩者。 • 無伺服器 AWS Lambda 函數版本。 您可以建立 CodeDeploy 應用程式、部署和部署群組，以便在階段中的部署動作中，在建立管道之前或在「建立管線」精靈中使用。 進一步了解： • AWS 入 DevOps 門指南 — 了解如何使用 CodePipeline 與 CodeDeploy 持續將 CodeCommit 儲存庫中的原始程式碼交付和部署到 Amazon EC2 執行個體。 • 簡單的管道演練（Amazon S3 存儲桶） • 簡單管道逐步解說 (CodeCommit 儲存庫) • 四階段管道教程

AWS 無伺服器應用模型	AWS 無伺服器應用程式模型 (AWS SAM) 是定義無伺服器應用程式的模型。它擴展 AWS CloudFormation 到提供了一種簡化的方式來定義無伺服器應用程式所需的 AWS Lambda 函數、Amazon API Gateway 和 Amazon DynamoDB 表格。如果您已經使用 AWS SAM，您可以新增部署喜好設定以開始 CodeDeploy 使用，以管理 AWS Lambda 應用程式部署期間流量轉移的方式。 如需詳細資訊，請參閱AWS 無伺服器應用程式模型。
Elastic Load Balancing	CodeDeploy 支援 E lastic Load Balancing，這項服務可將傳入的應用程式流量分配到多個 Amazon EC2 執行個體。 對於 CodeDeploy 部署，負載平衡器也可以防止流量在尚未準備就緒、目前部署到或不再需要作為環境的一部分時，將流量路由傳送到執行個體。 進一步了解： • Integrating CodeDeploy with Elastic Load Balancing

主題

- CodeDeploy 與 Amazon EC2 Auto Scaling 集成
- 與 Elastic CodeDeploy Load Balancing 整合

CodeDeploy 與 Amazon EC2 Auto Scaling 集成

CodeDeploy 支援 Amazon EC2 自 Auto Scaling，這是一項可根據您定義的條件自動啟動 Amazon EC2 執行個體的 AWS 服務。這些條件可能包括 CPU 使用率、磁碟讀取或寫入，或輸入或輸出網路流量的指定時間間隔內超過限制。Amazon EC2 Auto Scaling 會在不再需要執行個體時終止執行個

體。如需詳細資訊，請參閱什麼是 Amazon EC2 Auto Scaling？ 在 Amazon EC2 Auto Scaling 用戶指南中。

當新的 Amazon EC2 執行個體作為 Amazon EC2 自動擴展群組的一部分啟動時， CodeDeploy 可以自動將您的修訂部署到新執行個體。您也可以 CodeDeploy使用使用 Elastic Load Balancing 負載平衡器註冊的 Amazon EC2 Auto Scaling 執行個體協調部署。如需詳細資訊，請參閱 Integrating CodeDeploy with Elastic Load Balancing 及 在 Elastic Load Balancing 中為 CodeDeploy Amazon EC2 部署設定負載平衡器。

> **Note**
>
> 如果將多個部署群組與單一 Amazon EC2 Auto Scaling 群組建立關聯，可能會遇到問題。如果部署失敗，例如，執行個體會開始關閉，但其他執行中的部署需要一個小時才會逾時。如需詳細資訊，請參閱避免將多個部署群組與單一 Amazon EC2 Auto Scaling 群組建立關聯和引擎蓋之下： CodeDeploy 和 Amazon EC2 Auto Scaling 整合。

主題

- 將 CodeDeploy 應用程式部署到 Amazon EC2 Auto Scaling 群組
- 在 Auto Scaling 擴充事件期間啟用終止部署
- 亞馬遜 EC2 Auto Scaling 如何與 CodeDeploy
- 將自訂 AMI 與 CodeDeploy Amazon EC2 自 Auto Scaling 搭配使用

將 CodeDeploy 應用程式部署到 Amazon EC2 Auto Scaling 群組

若要將 CodeDeploy 應用程式修訂部署到 Amazon EC2 Auto Scaling 群組：

1. 建立或尋找可讓 Amazon EC2 Auto Scaling 群組與 Amazon S3 搭配使用的 IAM 執行個體設定檔。如需詳細資訊，請參閱 步驟 4：為您的 Amazon EC2 執行個體建立 IAM 執行個體設定檔。

   > **Note**
   >
   > 您也可以使用 CodeDeploy 將修訂從 GitHub 儲存庫部署到 Amazon EC2 Auto Scaling 群組。雖然 Amazon EC2 執行個體仍需要 IAM 執行個體設定檔，但設定檔不需要任何其他許可即可從 GitHub 儲存庫部署。

2. 建立或使用 Amazon EC2 Auto Scaling 群組，並在啟動組態或範本中指定 IAM 執行個體設定檔。如需詳細資訊，請參閱在 Amazon EC2 執行個體上執行之應用程式的 IAM 角色。

3. 建立或尋找允許 CodeDeploy 建立包含 Amazon EC2 Auto Scaling 群組的部署群組的服務角色。
4. 透過指定 Amazon EC2 Auto Scaling 群組名稱、服務角色和其他一些選項來建立部署群組。CodeDeploy如需詳細資訊，請參閱 建立就地部署的部署群組 (主控台) 或 建立就地部署的部署群組 (主控台) 。
5. 用於 CodeDeploy 將修訂部署到包含 Amazon EC2 Auto Scaling 群組的部署群組。

如需詳細資訊，請參閱 教學課程：用 CodeDeploy 於將應用程式部署到 Auto Scaling 群組。

在 Auto Scaling 擴充事件期間啟用終止部署

終止部署是一種部 CodeDeploy 署類型，會在發生 Auto Scaling 縮放事件時自動啟動。 CodeDeploy 在 Auto Scaling 服務終止執行個體之前，立即執行終止部署。在終止部署期間， CodeDeploy不會部署任何東西。相反地，它會產生生命週期事件，您可以連接到自己的指令碼，以啟用自訂關機功能。例如，您可以將`ApplicationStop`生命週期事件掛接到指令碼，該指令碼會在執行個體終止之前正常關閉應用程式。

若要取得終止部署期間 CodeDeploy 產生的生命週期事件清單，請參閱〈〉生命週期事件掛接可。

如果終止部署因任何原因而失敗， CodeDeploy 將允許執行個體終止繼續。這表示即 CodeDeploy使未執行完整的生命週期事件集 (或任何) 完成，執行個體仍會關閉。

如果您未啟用終止部署，Auto Scaling 服務仍會在擴充事件發生時終止 Amazon EC2 執行個體，但不 CodeDeploy 會產生生命週期事件。

> **Note**
>
> 無論您是否啟用終止部署，如果 Auto Scaling 服務在 CodeDeploy 部署進行時終止 Amazon EC2 執行個體，則 Auto Scaling 產生的生命週期事件和 CodeDeploy 服務之間可能會出現競爭情形。例如，生命`Terminating`週期事件 (由 Auto Scaling 服務產生) 可能會覆寫`ApplicationStart`事件 (由部 CodeDeploy 署產生)。在這個案例中，您可能會遇到 Amazon EC2 執行個體終止或 CodeDeploy 部署失敗的情況。

啟用執行終止部署 CodeDeploy 的步驟

- 建立或更新部署群組時，選取「將終止勾點新增至 Auto Scaling 群組」核取方塊。如需指示，請參閱建立就地部署的部署群組 (主控台)、或為EC2/內部部署藍/綠部署建立部署群組 (主控台)。

啟用此核取方塊會 CodeDeploy 將 Auto Scaling 生命週期勾點安裝到您在建立或更新 CodeDeploy 部署群組時指定的 Auto Scaling 群組。此勾點稱為終止勾點，並啟用終止部署。

安裝終止勾點之後，縮放 (終止) 事件會展開如下：

1. Auto Scaling 服務（或簡稱為 Auto Scaling）確定需要發生擴展事件，並聯繫 EC2 服務以終止 EC2 實例。
2. EC2 服務會開始終止 EC2 執行個體。執行個體會移至`Terminating`狀態，然後進入`Terminating:Wait`狀態。
3. 在期間`Terminating:Wait`，Auto Scaling 會執行附加至 Auto Scaling 群組的所有生命週期掛接，包括由安裝的終止勾點 CodeDeploy。
4. 終止勾點會將通知傳送至由輪詢的 Amazon SQS 佇列。 CodeDeploy
5. 收到通知後，會 CodeDeploy 剖析訊息、執行一些驗證，並執行終止部署。
6. 執行終止部署時，每五分鐘會將活動訊號 CodeDeploy 傳送至 Auto Scaling，讓其知道執行個體仍在使用中。
7. 到目前為止，EC2 實例仍處於狀`Terminating:Wait`態（如果啟用了 Auto Scaling 組暖池，則可能是`Warmed:Pending:Wait`狀態）。
8. 部署完成時，無論終止部署是成功還是失敗，都會向 `CONTINUE` EC2 終止程序 CodeDeploy 指示 Auto Scaling。

亞馬遜 EC2 Auto Scaling 如何與 CodeDeploy

當您建立或更新 CodeDeploy 部署群組以包含 Auto Scaling 群組時，請使用 CodeDeploy 服務角色 CodeDeploy存取 Auto Scaling 群組，然後將 Auto Scaling 生命週期掛接安裝到 Auto Scaling 群組中。

> **Note**
>
> Auto Scaling 生命週期掛鉤 CodeDeploy 與本指南中所產生並說明AppSpec 「掛鉤」部分的生命週期事件 (也稱為生命週期事件掛接) 不同。

CodeDeploy 安裝的「Auto Scaling」生命週期掛鉤為：

- 啟動勾點 — 此勾點通 CodeDeploy 知 Auto Scaling 向外延展事件正在進行中，且 CodeDeploy 需要啟動啟動部署。

在啟動部署期間， CodeDeploy：

- 將應用程式的修訂版部署到向外延展的執行個體。
- 產生生命週期事件以指示部署進度。您可以將這些生命週期事件連結至您自己的指令碼，以啟用自訂啟動功能。如需詳細資訊，請參閱中的表格生命週期事件掛接可。

啟動勾點和關聯的啟動部署永遠處於啟用狀態，且無法關閉。

- 終止勾點 — 此選用勾點通 CodeDeploy 知 Auto Scaling 擴充事件正在進行中，且 CodeDeploy 需要啟動終止部署。

  在終止部署期間， CodeDeploy 會產生生命週期事件以指出執行個體關閉的進度。如需詳細資訊，請參閱 在 Auto Scaling 擴充事件期間啟用終止部署。

主題

- CodeDeploy安裝生命週期掛鉤後，如何使用它們？
- 如何 CodeDeploy 命名 Amazon EC2 Auto Scaling 組
- 自訂生命週期掛接事件的執行順序
- 部署期間的向外延展事件
- 部署期間的擴充事件
- AWS CloudFormation cfn-init 腳本中的事件順序

CodeDeploy安裝生命週期掛鉤後，如何使用它們？

安裝啟動和終止生命週期掛接後，它們會分別 CodeDeploy 在 Auto Scaling 群組向外延展和縮放事件期間使用。

向外延展 (啟動) 事件展開如下：

1. Auto Scaling 服務（或簡稱為 Auto Scaling）確定需要發生向外擴展事件，並聯繫 EC2 服務以啟動新的 EC2 實例。
2. EC2 服務會啟動新的 EC2 執行個體。執行個體會移至`Pending`狀態，然後進入`Pending:Wait`狀態。
3. 在期間`Pending:Wait`，Auto Scaling 會執行附加至 Auto Scaling 群組的所有生命週期掛鉤，包括由安裝的啟動掛接 CodeDeploy。
4. 啟動勾點會將通知傳送至由輪詢的 Amazon SQS 佇列。 CodeDeploy

5. 收到通知後，會 CodeDeploy 剖析訊息、執行一些驗證，然後啟動啟動部署。
6. 啟動部署正在執行時，每五分鐘會將活動訊號 CodeDeploy 傳送至 Auto Scaling，讓其知道執行個體仍在使用中。
7. 到目前為止，EC2 實例仍處於狀`Pending:Wait`態。
8. 部署完成時，根據部署成功還是`CONTINUE`失敗，向 Auto Scaling CodeDeploy 指示或 EC2 啟動程序。`ABANDON`
   - 如果 CodeDeploy 指示`CONTINUE`，Auto Scaling 會繼續啟動程序，或等待其他掛接程序完成，或是先將執行個體置於`InService`狀態。`Pending:Proceed`
   - 如果 CodeDeploy 指示`ABANDON`，Auto Scaling 會終止 EC2 執行個體，並在需要時重新啟動啟動程序以滿足所需的執行個體數量，如 Auto Scaling 所需容量設定中所定義。

縮放 (終止) 事件展開如下：

請參閱在 Auto Scaling 擴充事件期間啟用終止部署。

如何 CodeDeploy 命名 Amazon EC2 Auto Scaling 組

在 EC2 /內部部署計算平台上進行藍/綠部署期間，您有兩個選項可將執行個體新增至替換 (綠色) 環境：

- 使用您手動建立或現有的執行個體。
- 使用您指定的 Amazon EC2 Auto Scaling 群組中的設定，在新的 Amazon EC2 Auto Scaling 群組中定義和建立執行個體。

如果您選擇第二個選項，請為您 CodeDeploy 佈建新的 Amazon EC2 Auto Scaling 群組。它使用以下慣例來為群組命名：

```
CodeDeploy_deployment_group_name_deployment_id
```

例如，如果具有 ID 的部署部署名為的部署群組`alpha-deployments`，則`10`會命名為已佈建的 Amazon EC2 Auto Scaling 群組。`CodeDeploy_alpha-deployments_10`如需詳細資訊，請參閱 為 EC2/內部部署藍/綠部署建立部署群組 (主控台) 及 GreenFleetProvisioningOption。

自訂生命週期掛接事件的執行順序

您可以將自己的生命週期掛鉤新增到 CodeDeploy 部署的 Amazon EC2 Auto Scaling 群組。但是，與預設部署生命週期事件相關的執行順序無法 CodeDeploy 預先確定這些自訂生命週期掛接事件的

執行順序。例如，如果您將名為的自訂生命週期勾點新增`ReadyForSoftwareInstall`到 Amazon EC2 Auto Scaling 群組，您就無法事先知道它是要在第一個預設部署生命週期事件之前還是在最後一個 CodeDeploy預設部署生命週期事件之後執行。

若要了解如何將自訂生命週期勾點新增至 Amazon EC2 Auto Scaling 群組，請參閱 Amazon EC2 Auto Scaling 使用者指南中的新增生命週期勾點。

部署期間的向外延展事件

如果在部署進行時發生 Auto Scale-Out 擴充事件，則新執行個體將以先前部署的應用程式修訂版更新，而不是最新的應用程式修訂版本。如果部署成功，舊執行個體和新向外延展的執行個體將會裝載不同的應用程式修訂版本。若要使具有較舊版本的執行個體保持最新狀態，請 CodeDeploy自動啟動後續部署 (在第一個執行個體之後立即)，以更新所有過期的執行個體。如果您想要變更此預設行為，讓過時的 EC2 執行個體保留在舊版本中，請參閱Automatic updates to outdated instances。

如果您想在部署進行時暫停 Amazon EC2 Auto Scaling 向外擴展流程，可以透過`common_functions.sh`指令碼中用於負載平衡的設定來執行此操作。 CodeDeploy如果`HANDLE_PROCS=true`，下列 Auto Scaling 事件會在部署程序期間自動暫停：

- AZRebalance
- AlarmNotification
- ScheduledActions
- ReplaceUnhealthy

> **Important**
>
> 只有 CodeDeployDefault. OneAtATIME 部署組態支援此功能。

有關在使用 Amazon EC2 Auto Scaling 時使用`HANDLE_PROCS=true`以避免部署問題的詳細資訊，請參閱（詳見）中有關處理 AutoScaling 程序的aws-codedeploy-samples重要注意事項 GitHub。

部署期間的擴充事件

如果 Auto Scaling 群組在該 Auto Scaling 群組上進行 CodeDeploy 部署時開始擴展，則終止程序 (包括終止部署生命週期事件) 與終止執行個體上的其他 CodeDeploy 生命週期事件之間可能會出現爭用情形。 CodeDeploy 如果在所有 CodeDeploy 生命週期事件完成之前終止執行個體，則該特定執行個體的部署可能會失敗。此外，整體 CodeDeploy 部署可能會失敗，也可能不會失敗，具體取決於您在部署組態中設定健全狀況最低主機設定的方式而定。

AWS CloudFormation cfn-init 腳本中的事件順序

如果您在最新佈建的 Linux-based 執行個體上使用 `cfn-init` (或 `cloud-init`) 執行命令碼，您的部署可能失敗，除非您在執行個體啟動後嚴格控制事件發生順序。

這順序必須：

1. 新佈建的新執行個體啟動。
2. 所有 `cfn-init` 引導操作命令碼完成執行。
3. CodeDeploy 代理程式會啟動。
4. 將最新的應用程式修訂版部署到執行個體中。

如果未仔細控制事件順序，則 CodeDeploy 代理程式可能會在所有指令碼完成執行之前啟動部署。

若要控制事件的順序，請使用這些最佳實務：

- 透過指`cfn-init`令碼安裝 CodeDeploy 代理程式，並將其置於所有其他指令碼之後。
- 將 CodeDeploy 代理程式包含在自訂 AMI 中，並使用指`cfn-init`令碼啟動它，並將其置於所有其他指令碼之後。

若要取得有關使用的資訊`cfn-init`，請參閱《使AWS CloudFormation 用指南》中的 cfn-init。

將自訂 AMI 與 CodeDeploy Amazon EC2 自 Auto Scaling 搭配使用

在 Amazon EC2 自動擴展群組中啟動新的 Amazon EC2 執行個體時，您有兩個選項可以指定要使用的基礎 AMI：

- 您可以指定已安裝 CodeDeploy 代理程式的基本自訂 AMI。由於代理程式已安裝，因此此選項會比其他選項更快地啟動新的 Amazon EC2 執行個體。但是，此選項提供 Amazon EC2 執行個體初始部署失敗的可能性更大，尤其是在 CodeDeploy 代理程式過期的情況下。如果您選擇此選項，建議您定期更新基礎自訂 AMI 中的 CodeDeploy 代理程式。
- 您可以指定未安裝 CodeDeploy 代理程式的基礎 AMI，並在 Amazon EC2 Auto Scaling 群組中啟動每個新執行個體時安裝代理程式。雖然此選項啟動新的 Amazon EC2 執行個體的速度比其他選項慢，但它提供了執行個體初始部署成功的可能性更大。此選項使用最新版本的 CodeDeploy代理程式。

與 Elastic CodeDeploy Load Balancing 整合

在 CodeDeploy 部署期間，負載平衡器可防止網際網路流量在尚未就緒、目前部署至執行個體或不再需要做為環境的一部分時，將網際網路流量路由至執行個體。負載平衡器扮演的確切角色，卻取決於它是用於藍/綠部署或就地部署。

> **Note**
>
> 在藍/綠部署中必須使用 Elastic Load Balancing 負載平衡器，而在就地部署中則是選用的。

Elastic Load Balancing 類型

Elastic Load Balancing 提供三種類型的負載平衡器，可用於 CodeDeploy 部署：傳統負載平衡器、應用程式負載平衡器和網路負載平衡器。

Classic Load Balancer

在傳輸層 (TCP/SSL) 或應用程式層 (HTTP/HTTPS) 進行路由及負載平衡。它支持 VPC。

> **Note**
>
> Amazon ECS 部署不支援傳統負載平衡器。

Application Load Balancer

在應用程式層 (HTTP/HTTPS) 的路由及負載平衡並支援以路徑為基礎的路由。此類型的負載平衡器能夠將請求路由至虛擬私有雲端 (VPC) 中的每個 EC2 執行個體或容器執行個體。

> **Note**
>
> 應用程式負載平衡器目標群組必須具有 EC2 執行個體上部署和 Fargate 部署的`instance`目標類型`IP`為。如需詳細資訊，請參閱目標類型。

Network Load Balancer

根據從 TCP 封包標頭擷取的位址資訊 (而非封包內容)，在傳輸層 (TCP/UDP 第 4 層) 進行路由和負載平衡。Network Load Balancer 可以處理流量暴增、保留用戶端的來源 IP，並在負載平衡器生命週期中使用固定 IP。

若要深入了解 Elastic Load Balancing 器，請參閱下列主題：

- 什麼是 Elastic Load Balancing？
- 什麼是 Classic Load Balancer？
- 什麼是 Application Load Balancer？
- 什麼是 Network Load Balancer？

藍/綠部署

在 Elastic Load Balancing 負載平衡器後方重新路由傳送執行個體流量是 CodeDeploy藍/綠部署的基礎。

在藍/綠部署期間，根據您指定的規則，負載平衡器可讓流量路由到部署群組中的新執行個體，而這個群組也是部署最新應用程式修訂版 (替代環境) 的群組，然後將流量從之前的應用程式修訂版所執行的舊執行個體 (原始環境) 區隔開來。

在取代環境中的執行個體向一或多個負載平衡器註冊之後，原始環境中的執行個體會取消註冊，並在您選擇的情況下終止。

對於藍/綠部署，您可以在部署群組中指定一或多個傳統負載平衡器、應用程式負載平衡器目標群組或 Network Load Balancer 目標群組。您可以使用 CodeDeploy 主控台或 AWS CLI 將負載平衡器新增至部署群組。

如需有關藍/綠部署中的負載平衡器詳細資訊，請參閱下列主題：

- 在 Elastic Load Balancing 中為 CodeDeploy Amazon EC2 部署設定負載平衡器
- 建立藍色/綠色部署 (主控台) 的應用程式
- 為EC2/內部部署藍/綠部署建立部署群組 (主控台)

就地部署：

在就地部署期間，負載平衡器會防止網際網路流量路由至即將部署到的目標執行個體，並會在部署到該執行個體完成後，讓執行個體再次開始接受流量。

如果在就地部署期間未使用負載平衡器，網際網路流量仍會在部署過程中導向到執行個體。因此，您的客戶可能會遇到中斷、不完整或過期的 Web 應用程式。當您在就地部署中使用 Elastic Load Balancing 器時，部署群組中的執行個體會從負載平衡器中取消註冊、更新為最新的應用程式修訂版，然後在部署成功後重新向負載平衡器註冊為相同部署群組的一部分。 CodeDeploy 將等待最多 1 個小時，讓執行個體在負載平衡器之後變得健康狀態良好。如果負載平衡器在等待期間未將執行個體標示為狀 CodeDeploy 況良好，則根據部署組態，移至下一個執行個體或部署失敗。

對於就地部署，您可以指定一或多個傳統負載平衡器、Application Load Balancer 目標群組或 Network Load Balancer 目標群組。您可以指定負載平衡器做為部署群組組態的一部分，也可以使用提供的指令碼 CodeDeploy 來實作負載平衡器。

使用部署群組指定就地部署負載平衡器

若要將負載平衡器新增至部署群組，請使用主 CodeDeploy 控台或 AWS CLI. 針對就地部署之有關在部署群組中指定負載平衡器的詳細資訊，請參閱下列主題：

- 建立就地部署的應用程式 (主控台)
- 建立就地部署的部署群組 (主控台)
- 在 Elastic Load Balancing 中為 CodeDeploy Amazon EC2 部署設定負載平衡器

使用指令碼指定就地部署負載平衡器

使用以下程序中的步驟來使用部署生命週期指令碼，以針對就地部署設定負載平衡。

> **Note**
>
> 您應該使用 CodeDeployDefault. OneAtATime 部署組態僅當您使用指令碼為就地部署設定負載平衡器時。不支援並行執行，而且 CodeDeployDefault. OneAtATIME 設置可確保腳本的串行執行。如需部署組態的詳細資訊，請參閱使用中的部署組態 CodeDeploy。

在的 CodeDeploy 範例存放庫中 GitHub，我們提供可調整以使用 E CodeDeploy lastic Load Balancing 負載平衡器的指示和範例。這些儲存庫包含三個範例 script— `register_with_elb.sh`

deregister_from_elb.sh、和 common_functions.sh —，這些指令碼提供您所需要的所有程式碼。只需在這三個指令碼中編輯預留位置，然後從您的 appspec.yml 檔案參考這些指令碼。

若要在 CodeDeploy 使用 Elastic Load Balancing 負載平衡器註冊的 Amazon EC2 執行個體中設定就地部署，請執行以下操作：

1. 下載您要用於就地部署之負載平衡器類型的範例：
   - Classic Load Balancer
   - Application Load Balancer 或 Network Load Balancer (任一類型皆可使用相同的指令碼)
2. 確保您的每個目標 Amazon EC2 執行個體都已 AWS CLI 安裝。
3. 確保每個目標 Amazon EC2 執行個體都具有 IAM 執行個體設定檔，至少連接了彈性負載平衡 * 和自動擴展:* 許可。
4. 在您的應用程式原始碼目錄中包含了部署生命週期事件指令碼 (register_with_elb.sh、deregister_from_elb.sh 和 common_functions.sh)。
5. 在應用程式修訂版本的中，提供在事件期間執行指register_with_elb.sh令碼的指示，CodeDeploy 以及在ApplicationStart事件期間執行ApplicationStop指deregister_from_elb.sh令碼。appspec.yml
6. 如果執行個體是 Amazon EC2 Auto Scaling 群組的一部分，您可以略過此步驟。

   在 common_functions.sh 指令碼中：

   - 如果您使用的是 C lassic Load Balancer，請在中指定 Elastic Load Balancing 負載平衡器的名稱ELB_LIST=""，並對檔案中其他部署設定進行所需的任何變更。
   - 如果您使用的是「Ap plication Load Balancer」或「Network Load Balancer」，請在中指定「Elastic Load Balancing」目標群組名稱的名稱TARGET_GROUP_LIST=""，然後對檔案中的其他部署設定進行必要的變更。
7. 將應用程式的原始碼、appspec.yml 以及部署生命週期事件指令碼配套成一個應用程式修訂版，然後上傳修訂版。將修訂部署到 Amazon EC2 執行個體。在部署期間，部署生命週期事件指令碼會取消向負載平衡器註銷 Amazon EC2 執行個體，等待連線耗盡，然後在部署完成後向負載平衡器重新註冊 Amazon EC2 執行個體。

與合作夥伴的產品和服務整合

CodeDeploy 內建下列合作夥伴產品與服務的整合：

Ansible	如果您已經有了一組 Ansible 劇本，但只需要在某個地方運行它們，那麼 Ansible 的模板並 CodeDeploy 演示了幾個簡單的部署鉤子如何確保 Ansible 在本地部署實例上可用並運行劇本。如果您已經有建立和維護庫存的程序，您也可以使用 Ansible 模組來安裝和執行 CodeDeploy 代理程式。 進一步了解： • 安智和 CodeDeploy
阿特拉西亞 — 竹和比桶	Bamboo 的 CodeDeploy 任務會將包含 AppSpec 檔案的目錄壓縮為 .zip 檔案，將檔案上傳到 Amazon S3，然後根據應用程式中提供的組態開始部署。 CodeDeploy Atlassian Bitbucket 支援 CodeDeploy 可讓您根據需求，將程式碼直接從 Bitbucket UI 推送至 Amazon EC2 執行個體，直接傳送至您的任何部署群組。這表示在您更新 Bitbucket 儲存庫中的程式碼之後，您不必登入持續整合 (CI) 平台或 Amazon EC2 執行個體即可執行手動部署程序。 進一步了解： • 使用竹子的 CodeDeploy 任務 • 宣布阿特拉西亞比特桶支持 CodeDeploy
Chef	AWS 提供了兩個模板樣本，用於集成廚師和 CodeDeploy. 第一個是安裝並啟動 CodeDeploy 代理程式的 Chef 食譜。這使您可以在使用 Chef 時繼續管理主機基礎結構 CodeDeploy。第二個示例模板演示瞭如 CodeDeploy 何使用在每個節點上使用廚師獨奏協調食譜和食譜的運行。 進一步了解：

	• 廚師和 CodeDeploy
CircleCI	CircleCI 提供自動化測試和持續整合及部署工具集。在中建立 IAM 角色以搭配 CircleCI 使用，並在 AWS 圈子 .yml 檔案中設定部署參數之後，您可以使用 CircleCI 搭配建立應用程式修訂、將其上傳 CodeDeploy 到 Amazon S3 儲存貯體，然後啟動和監控您的部署。 進一步了解： • 使用 CircleCI 將應用程式部署到 AWS CodeDeploy
CloudBees	您可以使用可在 CloudBeesDEV @cloud 上使用的 CodeDeploy 詹金斯外掛程式作為建置後動作。例如，在持續交付管道結尾，您可以使用它來部署應用程式修訂版到您的伺服器機群。 進一步了解： • CodeDeploy 詹金斯插件現在可在 DEV @cloud
Codeship	您可以使用 Codehip 透過 CodeDeploy部署應用程式修訂版。您可以使用 Codehip 使用者介面新增 CodeDeploy 至分支的部署管線。 進一步了解： • 部署到 CodeDeploy • CodeDeploy 在代碼協調整合

GitHub	您可以使用 CodeDeploy 從GitHub儲存庫部署應用程式修訂版本。每當 GitHub 儲存庫中的原始程式碼發生變更時，您也可以從儲存庫觸發部署。 進一步了解： • CodeDeploy 與整合 GitHub • 教學課程：用 CodeDeploy 來部署應用程式 GitHub • GitHub 使用時自動部署 CodeDeploy
HashiCorp 領事	您可以使用開放原始碼 HashiCorp Consul 工具，在 CodeDeploy中部署應用程式時，確保應用程式環境的健全狀況和穩定性。您可以使用 Consul 註冊應用程式，以便在部署時被發現，將應用程式和節點置於維護模式，將它們從部署解除，如果目標執行個體運作狀態不佳時，即可停止部署。 進一步了解： • CodeDeploy 使用 HashiCorp 領事的部署
Jenkins	CodeDeploy 詹金斯插件為您的詹金斯項目提供了一個構建後步驟。成功建置後，它會壓縮工作區、上傳到 Amazon S3，然後開始新的部署。 進一步了解： • CodeDeploy詹金斯插件 • 設置詹金斯插件 CodeDeploy

Puppet Labs	AWS 提供木偶和 CodeDeploy. 的範例範本。第一個是安裝並啟動 CodeDeploy 代理程式的 Puppet 模組。這可讓您在使用 Puppet 時繼續管理主機基礎結構 CodeDeploy。第二個範例範本示範如 CodeDeploy 何使用在每個節點上使用無主控傀儡來協調模組的執行和資訊清單。 進一步了解： • 木偶及 CodeDeploy
SaltStack	您可以將SaltStack基礎架構與 CodeDeploy. 您可以使用該 CodeDeploy 模塊在您的手下上安裝和運行 CodeDeploy 代理，或者使用幾個簡單的部署掛鉤，您可以使用 CodeDeploy 來協調 Salt States 的運行。 進一步了解： • SaltStack 和 CodeDeploy
TeamCity	您可以使用 CodeDeploy Runner 外掛程式直接從部署應用程式 TeamCity。此外掛程式新增了一個 TeamCity 建置步驟，可準備應用程式修訂並上傳至 Amazon S3 儲存貯體、在 CodeDeploy應用程式中註冊修訂、建立部 CodeDeploy 署，以及等待部署完成。 進一步了解： • CodeDeploy 亞軍 (下載) • CodeDeploy 亞軍插件（文檔）
Travis CI	您可以將 Travis CI 配置為在成功構建 CodeDeploy 後觸發部署。 進一步了解： • 特拉維斯 CI 和 CodeDeploy 部署

主題

CodeDeploy 與整合 GitHub

CodeDeploy 支持 GitHub，基於 Web 的代碼託管和共享服務。 CodeDeploy 可以將存放在儲存 GitHub庫或 Amazon S3 儲存貯體中的應用程式修訂版部署到執 CodeDeploy 僅支 GitHub 援 EC2 /內部部署。

主題

部署 CodeDeploy 修訂來源 GitHub

若要將應用程式修訂版本從 GitHub 儲存庫部署至執行個體：

1. 建立與 CodeDeploy 您要部署的 Amazon EC2 執行個體類型相容的修訂版。

   若要建立相容的修訂版，請遵循 規劃修訂 CodeDeploy 和 將應用程式規格檔案新增至修訂 CodeDeploy 之中的說明。
2. 使用 GitHub 帳戶將修訂新增至 GitHub 儲存庫。

   若要建立 GitHub 帳戶，請參閱加入 GitHub。若要建立 GitHub 儲存庫，請參閱建立存放庫。
3. 您可以使用 CodeDeploy 主控台中的「建立部署」頁面或 AWS CLI create-deployment命令，將修訂版本從 GitHub 儲存區域建置到設定用於建置的目標執行處理。 CodeDeploy

   如果您想要呼叫create-deployment命令，必須先使用主控台的 [建立部署] 頁面，為指定的應用程式提供代表您偏好 GitHub帳戶進行互動的 CodeDeploy 權限。 GitHub 每個應用程式都只需要執行此步驟一次。

   若要瞭解如何使用「建立建置」頁面從 GitHub 儲存區域建置，請參閱使用建立部署 CodeDeploy。

   若要瞭解如何呼叫要從 GitHub 儲存庫部署的create-deployment命令，請參閱建立EC2/內部部署計算平台部署 (CLI)。

   若要瞭解如何準備執行個體以用於 CodeDeploy 部署，請參閱使用的例證 CodeDeploy。

如需詳細資訊，請參閱 教學課程：用 CodeDeploy 來部署應用程式 GitHub。

GitHub 行為與 CodeDeploy

主題

- GitHub 使用應用程式的驗證 CodeDeploy
- CodeDeploy 與私人和公共 GitHub 存儲庫的交互
- CodeDeploy 與組織管理的儲 GitHub 存庫互動
- CodePipeline使用自動部署 CodeDeploy

GitHub 使用應用程式的驗證 CodeDeploy

在您 CodeDeploy 授予互動權限之後 GitHub，該 GitHub 帳戶和應用程式之間的關聯就會儲存在中 CodeDeploy。您可以將應用程式連結至其他 GitHub 帳戶。您也可以撤銷與之互 CodeDeploy 動的權限 GitHub。

將 GitHub 帳戶連結至應用程式 CodeDeploy

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在瀏覽窗格中，展開 [部署]，然後選擇 [應用程式]。
3. 選擇您要連結至其他 GitHub帳戶的應用程式。
4. 如果您的應用程式沒有部署群組，請選擇 [建立部署群組] 以建立部署群組。如需詳細資訊，請參閱 建立部署群組 CodeDeploy。部署群組需在後續步驟選擇 Create deployment (建立部署)。
5. 從 Deployments (部署) 選擇 Create deployment (建立部署)。

   > **Note**
   >
   > 您不必建立新的部署。這是目前將不同 GitHub 帳戶鏈接到應用程序的唯一方法。

6. 在部署設定中，對於修訂類型，選擇我的應用程式儲存於 GitHub。
7. 執行以下任意一項：

- 若要為 AWS CodeDeploy 應用程式與 GitHub帳戶建立連線，請 GitHub 在個別的網頁瀏覽器標籤中登出。在GitHub 權杖名稱中，輸入識別此連線的名稱，然後選擇 [Connect 線至] GitHub。網頁會提示您授 CodeDeploy權與 GitHub 您的應用程式互動。繼續步驟 10。
- 若要使用已建立的連線，請在GitHub權杖名稱中選取其名稱，然後選擇 [Connect 線至] GitHub。繼續步驟 8。
- 若要建立與其他 GitHub 帳戶的連線，請 GitHub 在個別的網頁瀏覽器標籤中登出。在GitHub 權杖名稱中，輸入識別連線的名稱，然後選擇 [Connect 線至] GitHub。網頁會提示您授 CodeDeploy 權與 GitHub 您的應用程式互動。繼續步驟 10。

8. 如果您尚未登入 GitHub，請依照 [登入] 頁面上的指示，使用您要連結應用程式的 GitHub 帳戶登入。
9. 選擇授權應用程式。 GitHub CodeDeploy授與代表所選應用 GitHub 程式的已登入 GitHub 帳戶進行互動的權限。
10. 如果您不想要建立部署，請選擇 Cancel (取消)。

撤銷與之互動 CodeDeploy 的權限 GitHub

1. GitHub 使用您要撤銷 AWS CodeDeploy 權限之 GitHub 帳戶的認證登入。
2. 開啟 [GitHub 應用程式] 頁面，找出CodeDeployed授權的應用程式清單，然後遵循撤銷應用 GitHub 程式授權的程序。

CodeDeploy 與私人和公共 GitHub 存儲庫的交互

CodeDeploy 支持從私有和公共 GitHub存儲庫部署應用程序。當您代表您 CodeDeploy 授予存 GitHub 取權限時， CodeDeploy將擁有您 GitHub 帳戶可存取的所有私有 GitHub 儲存庫的讀寫存取權。但是， CodeDeploy 只能從 GitHub 存儲庫中讀取。它不會寫入您的任何私有 GitHub 存儲庫。

CodeDeploy 與組織管理的儲 GitHub 存庫互動

根據預設，由組織管理的 GitHub 儲存庫 (與您帳戶自己的私人或公開儲存庫相反) 不會授予對第三方應用程式的存取權，包括 CodeDeploy. 如果在中啟用組織的協力廠商應用程式限制， GitHub 而您嘗試從其 GitHub 儲存庫部署程式碼，您的部署將會失敗。您有兩個方式可以解決這個問題：

- 身為組織成員，您可以要求組織擁有者核准存取權 CodeDeploy。請求此存取權的步驟取決於您是否已授權 CodeDeploy 個人帳戶：
  - 如果您已授權存取您帳戶 CodeDeploy 中的權限，請參閱要求授權應用程式的組織核准。
  - 如果您尚未授權帳戶 CodeDeploy 中的存取權，請參閱要求組織核准第三方應用程式。

- 機構組織擁有者可以停用所有第三方應用程式的限制。如需詳細資訊，請參閱停用組織的第三方應用程式限制。

有關詳情，請參閱關於第三方應用程式限制。

CodePipeline使用自動部署 CodeDeploy

您可以在原始程式碼變更 CodePipeline 時觸發部署。如需詳細資訊，請參閱 CodePipeline。

來自社群的整合範例

下列各節提供部落格文章、文章和社群所提供範例的連結。

> **Note**
>
> 這些連結僅供參考，不應視為範例內容的完整清單或背書。 AWS 不負責內容或是外部內容的準確性。

部落格文章

- 自動化 CodeDeploy 佈建 AWS CloudFormation

  了解如何使用來佈建中的應用程 CodeDeploy 式部署 AWS CloudFormation。

  發佈日期：2016 年 1 月
- AWS Toolkit for Eclipse 與整合 CodeDeploy (第 1 部分)

  AWS Toolkit for Eclipse 與整合 CodeDeploy (第 2 部分)

  AWS Toolkit for Eclipse 與集成 CodeDeploy （ 第 3 部分 ）

  瞭解 Java 開發人員如何使用 Eclipse 的 CodeDeploy 外掛程式， AWS 直接從 Eclipse 開發環境部署 Web 應用程式。

  發布日期：2015 年 2 月
- GitHub 使用時自動部署 CodeDeploy

  了解如何使用從 GitHub CodeDeploy 到的自動部署來建立 end-to-end 管道，從原始檔控制到測試或生產環境。

發佈日期：2014 年 12 月

CodeDeploy 教程

本節包含一些教學課程，可協助您學習如何使用 CodeDeploy。

這些教學課程中的程序會針對儲存檔案的位置 (例如 c:\temp) 以及要提供給值區、子資料夾或檔案的名稱 (例如，Codedeploydemobucket 和 CodeDeployDemo-EC2-Trust.JSON 分別) 提供建議，但您不需要使用它們。 HelloWorldApp執行程序時，請務必替代檔案位置和名稱。

主題

- 教學課程：部署 WordPress 至 Amazon EC2 執行個體 (Amazon Linux 或紅帽企業 Linux、macOS 或 Unix)
- 教程：部署一個「你好，世界！」 應用程序 CodeDeploy（視窗服務器）
- 教學課程：使用 CodeDeploy (Windows 伺服器、Ubuntu 伺服器或 RHEL) 將應用程式部署到內部部署執行個體
- 教學課程：用 CodeDeploy 於將應用程式部署到 Auto Scaling 群組
- 教學課程：用 CodeDeploy 來部署應用程式 GitHub
- 教學課程：將應用程式部署到 Amazon ECS
- 教學課程：透過驗證測試部署 Amazon ECS 服務
- 教學課程：使用 CodeDeploy 和 AWS 無伺服器應用程式模型部署更新的 Lambda 函數

教學課程：部署 WordPress 至 Amazon EC2 執行個體 (Amazon Linux 或紅帽企業 Linux、macOS 或 Unix)

在本教學中，您將以 PHP 和 MySQL 為基礎的開放原始碼部落格工具和內容管理系統，部署 WordPress到執行 Amazon Linux 或 RHEL (RHEL) 的單一亞馬遜 EC2 執行個體。

不是您想找的內容嗎？

- 若要改為練習部署到執行 Windows 伺服器的 Amazon EC2 執行個體，請參閱教程：部署一個「你好，世界！」 應用程序 CodeDeploy（視窗服務器）。
- 若要練習部署到現場部署執行個體而非 Amazon EC2 執行個體，請參閱教學課程：使用 CodeDeploy (Windows 伺服器、Ubuntu 伺服器或 RHEL) 將應用程式部署到內部部署執行個體。

本教學課程的步驟是從執行 Linux、macOS 或 Unix 的本機開發機器的角度來呈現。雖然您可以在執行 Windows 的本機電腦上完成其中大部分的步驟，但是您必須調整步驟以涵蓋命令，例如 chmod 和 wget，以及應用程式，例如 SED，還有目錄路徑，例如 `/tmp`。

開始此教學課程之前，您必須先完成 開始使用 CodeDeploy 中的先決條件。其中包括設定使用者、安裝或升級 AWS CLI，以及建立 IAM 執行個體設定檔和服務角色。

主題

步驟 1：啟動並設定 Amazon Linux 或紅帽企業 Linux Amazon EC2 執行個體

若要使 WordPress 用部署應用程式 CodeDeploy，您需要一個執行 Amazon Linux 或 RHEL (RHEL) 的亞馬遜 EC2 執行個體。Amazon EC2 執行個體需要允許 HTTP 連線的新輸入安全規則。成功部署後，需要此規則才能在瀏覽器中檢視 WordPress 頁面。

請遵循中的說明進行創建一個 Amazon EC2 實例 CodeDeploy 當您到達有關將 Amazon EC2 執行個體標籤指派給執行個體的指示中的部分時，請務必指定的標籤鍵**Name**和的標籤值**CodeDeployDemo**。(如果您指定不同的標籤鍵或標籤值，步驟 4：部署 WordPress應用程式中的說明可能會產生非預期的結果)。

按照指示啟動 Amazon EC2 執行個體後，請返回此頁面並繼續下一節。請勿繼續執建立應用程式 CodeDeploy行下一步。

Connect 到您的 Amazon Linux 或 RHEL Amazon EC2 實例

啟動新的 Amazon EC2 執行個體後，請按照下列指示練習連線到該執行個體。

1. 使用命ssh令（或像 PuTTY 這樣具有 SSH 功能的終端模擬器）連接到您的 Amazon Linux 或 RHEL Amazon EC2 實例。您需要執行個體的公有 DNS 地址，以及啟動 Amazon EC2 執行個體時使用的 key pair 的私密金鑰。如需詳細資訊，請參閱 Connect 至執行個體。

例如，如果公有 DNS 地址是**ec2-01-234-567-890.compute-1.amazonaws.com**，而用於 SSH 存取的 Amazon EC2 執行個體 key pair 已命名**codedeploydemo.pem**，您可以輸入：

```
ssh -i /path/to/codedeploydemo.pem ec2-
user@ec2-01-234-567-890.compute-1.amazonaws.com
```

將.pem檔案路徑和範例 DNS 位址取代/path/to/codedeploydemo.pem為您的 Amazon Linux 或 RHEL 亞馬遜 EC2 執行個體的地址。

> **Note**
>
> 如果您收到的錯誤是有關您的金鑰檔案許可過於開放，您需要限制其許可，使其只能存取目前使用者 (您)。例如，使用 Linux、macOS 或 Unix 上的chmod指令，輸入：

```
chmod 400 /path/to/codedeploydemo.pem
```

2. 登入後，您會看到 Amazon EC2 執行個體的 AMI 橫幅。對於 Amazon Linux，它應該是這樣的：

```
       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|
```

3. 您現在可以登出執行中的 Amazon EC2 執行個體。

> **Warning**
>
> 請勿停止或終止亞馬遜 EC2 執行個體。否則，將 CodeDeploy 無法部署到它。

新增允許 HTTP 流量傳入您的 Amazon Linux 或 RHEL 亞馬 Amazon EC2 執行個體的輸入規則

下一步會確認 Amazon EC2 執行個體具有開放的 HTTP 連接埠，因此您可以在瀏覽器中查看已部署 WordPress 應用程式的首頁。

1. 登入 AWS Management Console 並開啟 Amazon EC2 主控台，網址為 https://console.aws.amazon.com/ec2/。
2. 選擇執行個體，然後選擇您的執行個體。
3. 在 [說明] 索引標籤的 [安全性群組] 下，選擇 [檢視輸入規則]

   您應該會看到安全性群組中的規則清單，如下所示：

   ```
   Security Groups associated with i-1234567890abcdef0
    Ports     Protocol     Source     launch-wizard-N
    22        tcp          0.0.0.0/0          #
   ```

4. 在安全群組下，選擇 Amazon EC2 執行個體的安全群組。其可能被命名為 **launch-wizard-*N***。名稱中的 ***N*** 是一個您的執行個體建立時指派到安全群組的數字。

   選擇 Inbound (傳入) 標籤。如果執行個體的安全性群組設定正確，您應該會看到具有下列值的規則：

   - 類型：HTTP
   - Protocol (通訊協定)：TCP
   - Port Range (連接埠範圍)：80
   - 資料來源:0.0.0/0
5. 如果您沒有看到具有這些值的規則，請使用將規則新增至安全性群組中的程序，將其新增至新的安全性規則。

步驟 2：設定要部署到 Amazon Linux 或 RHEL 亞馬遜 EC2 執行個體的來源內容

現在，您可以設定應用程式的來源內容，以將某個項目部署至執行個體。

主題

- 獲取源代碼
- 創建腳本以運行您的應用程序
- 新增應用程式規格檔案

獲取源代碼

在本教學課程中，您將內 WordPress 容發佈平台從開發機器部署到目標 Amazon EC2 執行個體。若要取得 WordPress 原始程式碼，您可以使用內建的命令列呼叫。或者，如果您已於開發電腦上安裝 Git，可以改為使用該項目。

對於這些步驟，我們假設您將 WordPress 原始程式碼的複本下載到開發電腦上的/tmp目錄中。(您可以選擇想要的任何目錄，但請記得將這些步驟中指定的所有 /tmp 都替代為您的位置)。

選擇下列兩個選項之一，將 WordPress 來源檔案複製到您的開發電腦。第一個選項使用內建命令列呼叫。第二個選項使用 Git。

主題

- 獲取 WordPress 源代碼的副本（內置命令行調用）
- 若要取得 WordPress 原始程式碼 (Git) 的複本

獲取 WordPress 源代碼的副本（內置命令行調用）

1. 呼叫命wget令，將 WordPress 原始程式碼複本 (做為 .zip 檔案) 下載至目前目錄：

   ```
   wget https://github.com/WordPress/WordPress/archive/master.zip
   ```

2. 呼叫 unzip、mkdir、cp 和 rm 命令，以：

   - 將 master .zip 檔案解壓縮至 /tmp/WordPress_Temp 目錄 (資料夾)。
   - 將其解壓縮的內容複製至 /tmp/WordPress 目標資料夾。
   - 刪除暫時 /tmp/WordPress_Temp 資料夾和 master 檔案。

   一次執行一個命令：

   ```
   unzip master -d /tmp/WordPress_Temp
   ```

   ```
   mkdir -p /tmp/WordPress
   ```

   ```
   cp -paf /tmp/WordPress_Temp/WordPress-master/* /tmp/WordPress
   ```

```
rm -rf /tmp/WordPress_Temp
```

```
rm -f master
```

這會在資料夾中留下一組乾淨的 WordPress 原始程式碼檔/tmp/WordPress案。

若要取得 WordPress 原始程式碼 (Git) 的複本

1. 在開發電腦上，下載並安裝 Git。
2. 在 /tmp/WordPress 資料夾中，呼叫 git init 命令。
3. 調用命git clone令克隆公共 WordPress存儲庫，在/tmp/WordPress目標文件夾中製作自己的副本：

```
git clone https://github.com/WordPress/WordPress.git /tmp/WordPress
```

這會在資料夾中留下一組乾淨的 WordPress 原始程式碼檔/tmp/WordPress案。

創建腳本以運行您的應用程序

接下來，在目錄中創建一個文件夾和腳本。 CodeDeploy 使用這些指令碼在目標 Amazon EC2 執行個體上設定和部署應用程式修訂。您可以使用任何文字編輯器來建立指令碼。

1. 在 WordPress 源代碼副本中創建一個腳本目錄：

```
mkdir -p /tmp/WordPress/scripts
```

2. 在 /tmp/WordPress/scripts 中，建立 install_dependencies.sh 檔案。在檔案中新增下列各行。此 install_dependencies.sh 指令碼會安裝 Apache、MySQL 和 PHP。它還會新增 PHP 的 MySQL 支援。

```
#!/bin/bash
sudo amazon-linux-extras install php7.4
sudo yum install -y httpd mariadb-server php
```

3. 在 /tmp/WordPress/scripts 中，建立 start_server.sh 檔案。在檔案中新增下列各行。此 start_server.sh 指令碼會啟動 Apache 和 MySQL。

```
#!/bin/bash
systemctl start mariadb.service
systemctl start httpd.service
systemctl start php-fpm.service
```

4. 在 /tmp/WordPress/scripts 中，建立 stop_server.sh 檔案。在檔案中新增下列各行。此 stop_server.sh 指令碼會停止 Apache 和 MySQL。

```
#!/bin/bash
isExistApp=pgrep httpd
if [[ -n $isExistApp ]]; then
systemctl stop httpd.service
fi
isExistApp=pgrep mysqld
if [[ -n $isExistApp ]]; then
systemctl stop mariadb.service
fi
isExistApp=pgrep php-fpm
if [[ -n $isExistApp ]]; then
systemctl stop php-fpm.service

fi
```

5. 在 /tmp/WordPress/scripts 中，建立 create_test_db.sh 檔案。在檔案中新增下列各行。這個create_test_db.sh腳本使用 MySQL 來創建一個**test**數據庫 WordPress 以供使用。

```
#!/bin/bash
mysql -uroot <<CREATE_TEST_DB
CREATE DATABASE IF NOT EXISTS test;
CREATE_TEST_DB
```

6. 最後，在 /tmp/WordPress/scripts 中，建立 change_permissions.sh 指令碼。這用來在 Apache 中變更資料夾許可。

> **Important**
>
> 此指令碼已更新 /tmp/WordPress 資料夾的許可，讓任何人都可以寫入其中。這是必需的， WordPress 以便可以在期間寫入其數據庫步驟 5：更新和重新部署應用程式

WordPress。 WordPress 應用程式設定完成後，執行下列命令，將權限更新為更安全的設定：

```
chmod -R 755 /var/www/html/WordPress
```

```
#!/bin/bash
chmod -R 777 /var/www/html/WordPress
```

7. 給予所有指令碼可執行的許可。在命令列輸入：

```
chmod +x /tmp/WordPress/scripts/*
```

新增應用程式規格檔案

接下來，新增應用程式規格AppSpec 檔案 (檔案)，YAML 格式的檔案可用 CodeDeploy 於：

- 將應用程式修訂版中的來源檔案對應到目標 Amazon EC2 執行個體上的目的地。
- 指定已部署檔案的自訂許可。
- 指定要在部署期間在目標 Amazon EC2 執行個體上執行的指令碼。

檔 AppSpec 案必須命名`appspec.yml`。它必須放置在應用程式原始碼的根目錄中。在此教學課程中，根目錄是 `/tmp/WordPress`。

使用文字編輯器，建立名為 `appspec.yml` 的檔案。在檔案中新增下列各行：

```
version: 0.0
os: linux
files:
  - source: /
    destination: /var/www/html/WordPress
hooks:
  BeforeInstall:
    - location: scripts/install_dependencies.sh
      timeout: 300
```

```
    runas: root
AfterInstall:
  - location: scripts/change_permissions.sh
    timeout: 300
    runas: root
ApplicationStart:
  - location: scripts/start_server.sh
  - location: scripts/create_test_db.sh
    timeout: 300
    runas: root
ApplicationStop:
  - location: scripts/stop_server.sh
    timeout: 300
    runas: root
```

CodeDeploy 使用此 AppSpec 檔案將開發機器上/tmp/WordPress資料夾中的所有檔案複製到目標 Amazon EC2 執行個體上的/var/www/html/WordPress資料夾。root在部署期間， CodeDeploy 在部署生命週期期間的指定事件 (例如**BeforeInstall**和)，在目標 Amazon EC2 執行個體上的/var/www/html/WordPress/scripts資料夾中執行指定的指令碼**AfterInstall**。如果這些指令碼執行時間超過 300 秒 (5 分鐘)，請 CodeDeploy 停止部署並將部署標示為失敗。

如需這些設定的詳細資訊，請參閱CodeDeploy AppSpec 檔案參考。

> **Important**
>
> 此檔案中每個項目之間的空格位置和數目十分重要。如果間距不正確，會 CodeDeploy 引發可能難以除錯的錯誤。如需更多詳細資訊，請參閱 AppSpec 檔案間距。

步驟 3：將您的 WordPress應用程式上傳到 Amazon S3

現在，您將準備源內容並將其上傳到 CodeDeploy 可以部署它的位置。下列指示說明如何佈建 Amazon S3 儲存貯體、準備儲存貯體的應用程式修訂版檔案、捆綁修訂版本的檔案，然後將修訂推送至儲存貯體。

> **Note**
>
> 雖然本教學課程未說明，但您可以使用 CodeDeploy 將應用程式從 GitHub 儲存庫部署到執行個體。如需詳細資訊，請參閱 CodeDeploy 與整合 GitHub。

主題

佈建 Amazon S3 儲存貯體

在 Amazon S3 中建立儲存容器或儲存貯體，或使用現有的儲存貯體。請確定您可以將修訂版上傳到儲存貯體，且部署中使用的 Amazon EC2 執行個體可以從儲存貯體下載修訂版本。

您可以使用 AWS CLI Amazon S3 控制台或 Amazon S3 API 來創建一個 Amazon S3 存儲桶。建立值區之後，請務必授予值區和 AWS 帳戶的存取權限。

> **Note**
>
> 所有 AWS 帳戶的儲存貯體名稱在 Amazon S3 中都必須是唯一的。如果您無法使用 **codedeploydemobucket**，請嘗試不同的儲存貯體名稱 (例如後接破折號和您的縮寫或其他唯一識別符的 **codedeploydemobucket**)。然後，請務必將在本教學中看到的所有 **codedeploydemobucket** 都替代為您的儲存貯體名稱。
>
> Amazon S3 儲存貯體必須在啟動目標 Amazon EC2 執行個體的相同 AWS 區域中建立。例如，如果您在美國東部 (維吉尼亞北部) 區域建立儲存貯體，則必須在美國東部 (維吉尼亞北部) 區域啟動目標 Amazon EC2 執行個體。

主題

若要建立 Amazon S3 儲存貯體 (CLI)

呼叫命mb令以建立名為的 Amazon S3 儲存貯體**codedeploydemobucket**：

```
aws s3 mb s3://codedeploydemobucket --region region
```

若要建立 Amazon S3 儲存貯體 (主控台)

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
2. 在 Amazon S3 主控台中，選擇「建立儲存貯體」。
3. 在 Bucket name (儲存貯體名稱) 方塊中，輸入儲存貯體的名稱。
4. 在 Region (區域) 清單中，選擇目標區域，然後選擇 Create (建立)。

授予權限給 Amazon S3 儲存貯體和 AWS 帳戶

您必須擁有上傳到 Amazon S3 儲存貯體的許可。您可以透過 Amazon S3 儲存貯體政策指定這些許可。例如，在下列 Amazon S3 儲存貯體政策中，使用萬用字元 (*) 可讓 AWS 帳戶111122223333將檔案上傳到 Amazon S3 儲存貯體中名為的任何目錄codedeploydemobucket：

```
{
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::codedeploydemobucket/*",
            "Principal": {
                "AWS": [
                    "111122223333"
                ]
            }
        }
    ]
}
```

若要檢視您的 AWS 帳戶 ID，請參閱尋找您的 AWS 帳戶 ID。

現在是驗證 Amazon S3 儲存貯體允許從每個參與的 Amazon EC2 執行個體下載請求的好時機。您可以透過 Amazon S3 儲存貯體政策進行指定。例如，在下列 Amazon S3 儲存貯體政策中，使用萬用字元 (*) 可讓任何具codedeploydemobucket有附加 IAM 執行個體設定檔包含 ARN arn:aws:iam::444455556666:role/CodeDeployDemo 的 Amazon EC2 執行個體從 Amazon S3 儲存貯體中的任何目錄下載檔案：

```
{
```

```
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::codedeploydemobucket/*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666:role/CodeDeployDemo"
                ]
            }
        }
    ]
}
```

如需如何產生和連接 Amazon S3 儲存貯體政策的詳細資訊，請參閱儲存貯體政策範例。

如需如何建立和附加 IAM 政策的詳細資訊，請參閱使用政策。

為值區準備應用程式的檔案

請確定 WordPress 應用程式檔 AppSpec 案、檔案和指令碼在您的開發電腦上進行組織，如下所示：

```
/tmp/
  |--WordPress/
      |-- appspec.yml
      |-- scripts/
      |    |-- change_permissions.sh
      |    |-- create_test_db.sh
      |    |-- install_dependencies.sh
      |    |-- start_server.sh
      |    |-- stop_server.sh
      |-- wp-admin/
      |    |-- (various files...)
      |-- wp-content/
      |    |-- (various files...)
      |-- wp-includes/
      |    |-- (various files...)
      |-- index.php
      |-- license.txt
      |-- readme.html
```

```
    |-- (various files ending with .php...)
```

將應用程序的文件捆綁到單個存檔文件中，然後推送歸檔文件

將 WordPress 應用程式檔案和 AppSpec 檔案捆綁到歸檔檔案中 (稱為應用程式修訂版)。

> **Note**
>
> 可能會向您收取下列作業的費用：在儲存貯體中存放物件，以及將應用程式修訂傳入和傳出儲存貯體。如需詳細資訊，請參閱 Simple Storage Service (Amazon S3) 定價。

1. 在開發電腦上，切換至檔案存放所在的資料夾：

   ```
   cd /tmp/WordPress
   ```

   > **Note**
   >
   > 如果您未切換至此資料夾，將會在目前的資料夾開始檔案綁定。例如，如果您的目前資料夾是 `/tmp`，而不是 `/tmp/WordPress`，則會開始綁定 `tmp` 資料夾中的檔案和子資料夾，而此資料夾可能不只包含 `WordPress` 子資料夾。

2. 呼叫 create-application 命令，向名為 **`WordPress_App`** 的新應用程式註冊：

   ```
   aws deploy create-application --application-name WordPress_App
   ```

3. 呼叫 CodeDeploy push 命令將檔案捆綁在一起、將修訂上傳到 Amazon S3，然後在一個動作中註冊有 CodeDeploy 關上傳修訂的資訊。

   ```
   aws deploy push \
     --application-name WordPress_App \
     --s3-location s3://codedeploydemobucket/WordPressApp.zip \
     --ignore-hidden-files
   ```

   此指令會將目前目錄中的檔案 (不包括任何隱藏檔案) 捆綁到一個名為的單一封存檔案中**`WordPressApp.zip`**，將修訂上載到**`codedeploydemobucket`**值區，並將有 CodeDeploy 關上載修訂的資訊註冊到值區。

步驟 4：部署 WordPress應用程式

現在，您可以部署上傳到 Amazon S3 的範例 WordPress 應用程式修訂版。您可以使用 AWS CLI 或主 CodeDeploy 控台部署修訂版並監視部署進度。成功部署應用程式修訂版之後，您要檢查結果。

主題

部署您的應用程式修訂 CodeDeploy

使用 AWS CLI 或主控台部署應用程式修訂版本。

主題

部署應用程式修訂 (CLI)

1. 部署需要部署群組。不過，在您建立部署群組之前，需要服務角色 ARN。服務角色是 IAM 角色，可為您提供服務權限。在這種情況下，服務角色 CodeDeploy 授予存取 Amazon EC2 執行個體的權限，以擴展 (讀取) 其 Amazon EC2 執行個體標籤。

   您應該已經遵循建立服務角色 (CLI) 中的說明，來建立服務角色。若要取得服務角色的 ARN，請參閱取得服務角色 ARN (CLI)。
2. 現在您已經擁有服務角色 ARN，請使用名為以下create-deployment-group命名的 Amazon EC2 標籤**CodeDeployDemo**和部署組態來建立名**WordPress_DepGroup**為的部署群組**WordPress_App**，與名為的應用程式相關聯的部署群組：**CodeDeployDefault.OneAtATime**

   ```
   aws deploy create-deployment-group \
     --application-name WordPress_App \
     --deployment-group-name WordPress_DepGroup \
     --deployment-config-name CodeDeployDefault.OneAtATime \
     --ec2-tag-filters Key=Name,Value=CodeDeployDemo,Type=KEY_AND_VALUE \
     --service-role-arn serviceRoleARN
   ```

> **Note**
>
> 該create-deployment-group命令支援建立觸發器，以便將 Amazon SNS 通知傳送給主題訂閱者有關部署和執行個體中指定事件的主題訂閱者。該命令還支援自動復原部署和設定警示以停止部署的選項，以便在符合 Amazon CloudWatch 警示中的監控閾值時停止部署。這些動作的指令不包括在本教學課程中。

3. 建立部署之前，部署群組中的執行個體必須已安裝 CodeDeploy 代理程式。您可以使用 AWS Systems Manager 與下列命令，從命令列安裝代理程式：

```
aws ssm create-association \
  --name AWS-ConfigureAWSPackage \
  --targets Key=tag:Name,Values=CodeDeployDemo \
  --parameters action=Install,name=AWSCodeDeployAgent \
  --schedule-expression "cron(0 2 ? * SUN *)"
```

此命令會在系統管理員狀態管理員中建立一個關聯，以安裝 CodeDeploy 代理程式，然後嘗試在每個星期天早上 2:00 進行更新。如需 CodeDeploy 代理程式的詳細資訊，請參閱使用 CodeDeploy 代理程式。如需有關 Systems Manager 的詳細資訊，請參閱什麼是 AWS Systems Manager。

4. 現在，請在名為 **codedeploydemobucket** 的儲存貯體中使用名為 **WordPressApp.zip** 的應用程式修訂，呼叫 create-deployment 命令來建立與名為 **WordPress_App** 之應用程式、名為 **CodeDeployDefault.OneAtATime** 之部署組態和名為 **WordPress_DepGroup** 之部署群組建立關聯的部署：

```
aws deploy create-deployment \
  --application-name WordPress_App \
  --deployment-config-name CodeDeployDefault.OneAtATime \
  --deployment-group-name WordPress_DepGroup \
  --s3-location bucket=codedeploydemobucket,bundleType=zip,key=WordPressApp.zip
```

部署應用程式修訂 (主控台)

1. 使用 CodeDeploy 主控台部署應用程式修訂版本之前，您需要服務角色 ARN。服務角色是 IAM 角色，可為您提供服務權限。在這種情況下，服務角色 CodeDeploy 授予存取 Amazon EC2 執行個體的權限，以擴展 (讀取) 其 Amazon EC2 執行個體標籤。

您應該已經遵循建立服務角色 (主控台) 中的說明，來建立服務角色。若要取得服務角色的 ARN，請參閱取得服務角色 ARN (主控台)。

2. 現在您已經擁有 ARN，請使用 CodeDeploy 主控台部署應用程式修訂版本：

   請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

3. 在瀏覽窗格中，展開 [部署]，然後選擇 [應用程式]。
4. 在應用程式清單中，選擇 [WordPress_App]。
5. 在 Deployment groups (部署群組) 標籤中，選擇 Create deployment group (建立部署群組)。
6. 在 Deployment group name (部署群組名稱) 中，輸入 **`WordPress_DepGroup`**。
7. 在 Deployment type (部署類型) 下，選擇 In-place deployment (就地部署)。
8. 在環境組態中，選取 Amazon EC2 執行個體。
9. 在 [代理程式組態使用] 中 AWS Systems Manager，保留預設值。
10. 在 Key (金鑰) 中，輸入 **`Name`**。
11. 在 Value (值) 中輸入 **`CodeDeployDemo`**。

    > **Note**
    > 輸入後**`CodeDeployDemo`**，相符執行個體下應該會出現 1，以確認 CodeDeploy 找到一個相符的 Amazon EC2 執行個體。

12. 在 [部署組態] 中，選擇CodeDeployDefault。 OneAtA 時間。
13. 在 Service role ARN (服務角色 ARN) 中，選擇服務角色 ARN，然後選擇 Create deployment group (建立部署群組)。
14. 選擇 Create deployment (建立部署)。
15. 在 Deployment group (部署群組) 中選擇 **`WordPress_DepGroup`**。
16. 在 [存放庫類型] 旁邊，選擇 [我的應用程式存放在 Amazon S3]。在修訂位置中，輸入先前上傳到 Amazon S3 的範例 WordPress 應用程式修訂版的位置。取得位置：

a. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。

b. 在值區清單中，選擇 codedeploydemobucket (或您上傳應用程式修訂版的值區名稱)。

c. 在物件清單中，選擇 WordPressApp.zip。

d. 在 Overview (概觀) 標籤上，將 Link (連結) 欄位的值複製至剪貼簿。

這看起來類似下述：

https://s3.amazonaws.com/codedeploydemobucket/WordPressApp.zip

e. 返回 CodeDeploy 主控台，然後在「修訂版本」位置貼上「連結」欄位值。

17. 如果說明無法偵測到檔案類型的訊息出現在 File type (檔案類型) 清單中，請選擇 .zip。
18. (選用) 在 Deployment description (部署說明) 方塊中，輸入註解。
19. 展開部署群組覆寫，然後從部署組態中選擇CodeDeployDefault。 OneAtA 時間。
20. 選擇 Start deployment (啟動部署)。新建立部署的相關資訊會顯示在 Deployments (部署) 頁面上。

監控和疑難排解您的部署

使用 AWS CLI 或主控台來監控部署並進行疑難排解。

主題

- 監控部署並進行疑難排解 (CLI)
- 監控和故障診斷部署 (主控台)

監控部署並進行疑難排解 (CLI)

1. 針對名為 **WordPress_App** 的應用程式和名為 **WordPress_DepGroup** 的部署群組呼叫 list-deployments 命令，來取得部署 ID：

```
aws deploy list-deployments --application-name WordPress_App --deployment-group-
name WordPress_DepGroup --query 'deployments' --output text
```

2. 使用部署 ID，呼叫 get-deployment 命令：

```
aws deploy get-deployment --deployment-id deploymentID --query
 'deploymentInfo.status' --output text
```

3. 此命令傳回部署的整體狀態。如果成功，值為 Succeeded。

   如果整體狀態為Failed，您可以呼叫list-deployment-instances和之類的命令get-deployment-instance來進行疑難排解。如需其他故障診斷選項，請參閱分析日誌檔案以調查執行個體的部署失敗。

監控和故障診斷部署 (主控台)

在 CodeDeploy 主控台的「部署」頁面上，您可以在「狀態」欄中監視部署的狀態。

若要取得部署的詳細資訊，特別是在 Status (狀態) 欄位中，有任何數值不是 Succeeded (成功) 的情況下，則可執行以下動作：

1. 在 Deployments (部署) 資料表中，選擇部署的名稱。部署失敗後，會顯示失敗原因的訊息。
2. 在 Instance activity (執行個體活動) 中，會顯示更多有關部署的資訊。部署失敗後，您可能可以判斷哪些 Amazon EC2 執行個體以及部署失敗的步驟。
3. 您可以使用View Instance Details中所述的這類技術，藉此執行其他疑難排解。您也可以分析 Amazon EC2 執行個體上的部署日誌檔。如需詳細資訊，請參閱 分析日誌檔案以調查執行個體的部署失敗。

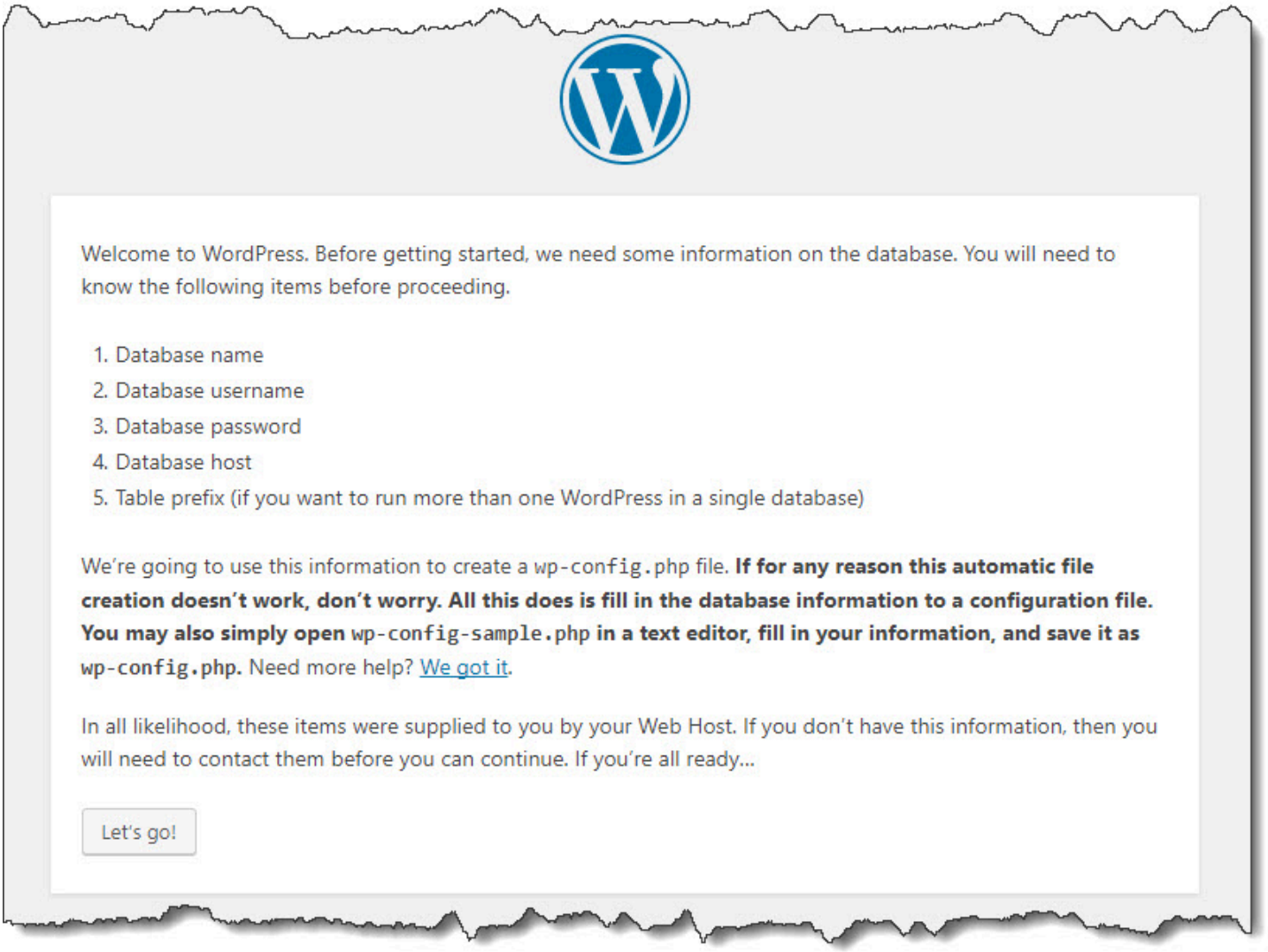
驗證您的部署

部署成功後，請確認您的 WordPress 安裝是否正常運作。使用 Amazon EC2 執行個體的公有 DNS 地址，然後使/WordPress用網頁瀏覽器檢視您的網站。若要取得公有 DNS 值，請在 Amazon EC2 主控台中選擇 Amazon EC2 執行個體，然後在「描述」索引標籤上尋找公用 DNS 的值。)

例如，如果您的 Amazon EC2 執行個體的公有 DNS 地址是**ec2-01-234-567-890.compute-1.amazonaws.com**，您可以使用下列 URL：

```
http://ec2-01-234-567-890.compute-1.amazonaws.com/WordPress
```

當您在瀏覽器中檢視網站時，您應該會看到類似下列內容的 WordPress 歡迎頁面：



如果您的 Amazon EC2 執行個體未將 HTTP 輸入規則新增至其安全群組，則不會顯示 WordPress 歡迎頁面。如果您看到訊息指出遠端伺服器沒有回應，請確定 Amazon EC2 執行個體的安全群組具有輸入規則。如需更多詳細資訊，請參閱 新增允許 HTTP 流量傳入您的 Amazon Linux 或 RHEL 亞馬 Amazon EC2 執行個體的輸入規則。

步驟 5：更新和重新部署應用程式 WordPress

現在您已成功部署應用程式修訂版，請在開發電腦上更新程式 WordPress 碼，然後使 CodeDeploy 用重新部署網站。之後，您應該會在 Amazon EC2 執行個體上看到程式碼變更。

主題

- 設置網 WordPress 站
- 修改網站

- 重新部署網站

設置網 WordPress 站

若要查看程式碼變更的影響，請完成 WordPress 網站的設定，讓您擁有完整功能的安裝。

1. 將網站的 URL 輸入 Web 瀏覽器中。該 URL 是 Amazon EC2 執行個體的公有 DNS 地址加上/WordPress擴充功能。對於此範例 WordPress 網站 (以及範例亞馬遜 EC2 執行個體公有 DNS 地址)，URL 為**http://ec2-01-234-567-890.compute-1.amazonaws.com/WordPress**。
2. 如果您尚未設定網站，則會顯示 WordPress 預設歡迎頁面。選擇 Let's go! (開始吧！)。
3. 若要使用預設的 MySQL 資料庫，請在資料庫組態頁面中輸入以下值：

   - 資料庫名稱：**test**
   - 使用者名稱：**root**
   - 密碼：保留空白。
   - 資料庫主機：**localhost**
   - 資料表字首：**wp_**

   選擇 Submit (提交) 以設定資料庫。
4. 繼續進行網站設定。在 [歡迎使用] 頁面上，填入您想要的任何值，然後選擇 [安裝] WordPress。當安裝完成後，您就可以登入您的儀表板。

> **Important**
>
> 在部署 WordPress 應用程式期間，**change_permissions.sh**指令碼會更新/tmp/WordPress資料夾的權限，讓任何人都可以寫入資料夾。現在是時候執行以下命令來限制許可，以便只有擁有者 (您) 可以寫入：
>
> ```
> chmod -R 755 /var/www/html/WordPress
> ```

修改網站

要修改 WordPress 網站，請轉到開發計算機上的應用程序的文件夾：

```
cd /tmp/WordPress
```

若要修改網站的一些顏色，請在 wp-content/themes/twentyfifteen/style.css 檔案中，使用文字編輯器或 sed 將 #fff 變更為 #768331。

在 Linux 或具有 GNU sed 的其他系統上，使用：

```
sed -i 's/#fff/#768331/g' wp-content/themes/twentyfifteen/style.css
```

在 macOS、Unix 或具有 BSD sed 的其他系統上，使用：

```
sed -i '' 's/#fff/#768331/g' wp-content/themes/twentyfifteen/style.css
```

重新部署網站

現在您已經修改了網站的程式碼，請使用 Amazon S3 並 CodeDeploy 重新部署網站。

如中所述，將變更捆綁並上傳到 Amazon S3 將應用程序的文件捆綁到單個存檔文件中，然後推送歸檔文件。(當您遵循這些說明時，請記住您不需要建立應用程式)。如以前一樣將相同的金鑰給予新的修訂 (**WordPressApp.zip**)。將其上傳到您之前建立的相同 Amazon S3 儲存貯體 (例如**codedeploydemobucket**)。

使用 AWS CLI、主 CodeDeploy 控台或 CodeDeploy API 重新部署網站。

主題

- 重新部署網站 (CLI)
- 重新部署網站 (主控台)

重新部署網站 (CLI)

呼叫 create-deployment 命令來根據新上傳的修訂版建立部署。使用名為 **WordPress_App**的應用程式、名為 **CodeDeployDefault.OneAtATime** 的部署組態、名為 **WordPress_DepGroup**的部署群組、名為 **WordPressApp.zip** 的修訂版 (在名為 **codedeploydemobucket**的儲存貯體中)：

```
aws deploy create-deployment \
 --application-name WordPress_App \
 --deployment-config-name CodeDeployDefault.OneAtATime \
```

```
  --deployment-group-name WordPress_DepGroup \
  --s3-location bucket=codedeploydemobucket,bundleType=zip,key=WordPressApp.zip
```

您可以檢查部署的狀態，如監控和疑難排解您的部署中所述。

重新部署網站後 CodeDeploy ，請在 Web 瀏覽器中重新造訪該網站，以確認顏色已變更。(您可能需要重新整理瀏覽器)。如果顏色已經變更，那麼恭喜！您已成功修改並重新部署該網站！

重新部署網站 (主控台)

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在瀏覽窗格中，展開 [部署]，然後選擇 [應用程式]。
3. 在應用程式清單中，選擇 [WordPress_App]。
4. 在 Deployment groups (部署群組) 標籤上，選擇 **WordPress_DepGroup**。
5. 選擇 Create deployment (建立部署)。
6. 請在 Create deployment (建立部署) 頁面上，執行以下操作：

   a. 在 Deployment group (部署群組) 中，選擇 **WordPress_DepGroup**。

   b. 在 [存放庫類型] 區域中，選擇 [我的應用程式存放在 Amazon S3]，然後將修訂的 Amazon S3 連結複製到修訂位置方塊中。尋找連結值：

      i. 在單獨的瀏覽器標籤中：

         登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。

         瀏覽並開啟程式碼部署 Demobucket，然後選擇您的修訂版。**WordPressApp.zip**

      ii. 如果在 Amazon S3 主控台中看不到「屬性」窗格，請選擇「屬性」按鈕。

      iii. 在「屬性」窗格中，將「連結」欄位的值複製到 CodeDeploy 主控台的「修訂版位置」方塊中。

   c. 如果出現無法偵測檔案類型的訊息，則請選擇 .zip (.zip)。

   d. 將 Deployment description (部署說明) 方塊留白。

e. 展開部署群組覆寫，然後從部署組態中選擇CodeDeployDefault。 OneAtA 時間。

f. 選擇 Start deployment (啟動部署)。新建立部署的相關資訊會顯示在 Deployments (部署) 頁面上。

g. 您可以檢查部署的狀態，如監控和疑難排解您的部署中所述。

重新部署網站後 CodeDeploy ，請在 Web 瀏覽器中重新造訪該網站，以確認顏色已變更。(您可能需要重新整理瀏覽器)。如果顏色已經變更，那麼恭喜！您已成功修改並重新部署該網站！

步驟 6：清理您的 WordPress 應用程式和相關資源

您現在已成功更新程 WordPress 式碼並重新部署網站。為了避免本教學中所建立的資源持續發生費用，您應該刪除：

- 任何 AWS CloudFormation 堆疊 (或終止任何 Amazon EC2 執行個體，如果您在以外建立它們 AWS CloudFormation)。
- 任何 Amazon S3 桶。
- CodeDeploy 中的 `WordPress_App` 應用程式。
- CodeDeploy 代理程式的 AWS Systems Manager 狀態管理員關聯。

您可以使用 AWS CLI、 AWS CloudFormation、Amazon S3、Amazon EC2 和 CodeDeploy 主控台或 AWS API 來執行清理。

主題

清除資源 (CLI)

1. 如果您在本教程中使用了我們的 AWS CloudFormation 模板，請對名為的堆棧調用delete-stack命令**`CodeDeployDemoStack`**。這將終止所有隨附的 Amazon EC2 執行個體，並刪除堆疊建立的所有隨附 IAM 角色：

```
aws cloudformation delete-stack --stack-name CodeDeployDemoStack
```

2. 若要刪除 Amazon S3 儲存貯體，請對名為的儲存貯體使用--recursive交換器呼叫rm命令**codedeploydemobucket**。這會刪除儲存貯體以及儲存貯體中的所有物件：

```
aws s3 rm s3://codedeploydemobucket --recursive --region region
```

3. 若要刪除 WordPress_App 應用程式，請呼叫 delete-application 命令。這也會刪除應用程式的所有相關聯部署群組記錄和部署記錄：

```
aws deploy delete-application --application-name WordPress_App
```

4. 若要刪除「Systems Manager 狀態管理員」關聯，請呼叫delete-association指令。

```
aws ssm delete-association --assocation-id association-id
```

您可以通過調用命###### ID。describe-association

```
aws ssm describe-association --name AWS-ConfigureAWSPackage --targets
 Key=tag:Name,Values=CodeDeployDemo
```

如果您在本教學中未使用 AWS CloudFormation 堆疊，請呼叫terminate-instances命令以終止您手動建立的任何 Amazon EC2 執行個體。提供要終止的 Amazon EC2 實例的 ID：

```
aws ec2 terminate-instances --instance-ids instanceId
```

清除資源 (主控台)

如果您在本教程中使用了我們的 AWS CloudFormation 模板，請刪除關聯的 AWS CloudFormation 堆棧。

1. 請登入 AWS Management Console 並開啟 AWS CloudFormation 主控台，網址為 https://console.aws.amazon.com/cloudformation。
2. 在「篩選條件」(Filter) 方塊中，輸入您先前建立的 AWS CloudFormation 堆疊名稱 (例如**CodeDeployDemoStack**)。
3. 選取堆疊名稱旁的方塊。在 Actions (動作) 選單中，選擇 Delete Stack (刪除堆疊)。

   AWS CloudFormation 刪除堆疊、終止所有隨附的 Amazon EC2 執行個體，並刪除所有隨附的 IAM 角色。

若要終止您在 AWS CloudFormation 堆疊外部建立的 Amazon EC2 執行個體：

1. 登入 AWS Management Console 並開啟 Amazon EC2 主控台，網址為 https://console.aws.amazon.com/ec2/。
2. 在 INSTANCES (執行個體) 清單中，選擇 Instances (執行個體)。
3. 在搜尋方塊中，輸入您要終止的 Amazon EC2 執行個體名稱 (例如，**CodeDeployDemo**)，然後按 Enter 鍵。
4. 選擇 Amazon EC2 實例名稱。
5. 在 Actions (動作) 選單中，指向 Instance State (執行個體狀態)，然後選擇 Terminate (終止)。出現提示時，選擇 Yes, Terminate (是，終止)。

為每個執行個體重複這些步驟。

要刪除 Amazon S3 存儲桶：

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在儲存貯體清單中，瀏覽並選擇您先前建立的 Amazon S3 儲存貯體的名稱 (例如**codedeploydemobucket**)。
3. 您必須先刪除其內容，才能刪除儲存貯體。選擇儲存貯體中的所有檔案，例如 **WordPressApp.zip**。在操作功能表中，選擇刪除。出現提示要您確認刪除時，選擇 OK (確定)。
4. 儲存貯體清空之後，您即可刪除儲存貯體。在儲存貯體清單中，選擇儲存貯體的資料列 (但不是儲存貯體名稱)。選擇 Delete bucket (刪除儲存貯體)，然後在出現確認提示時，選擇 OK (確定)。

若要從中刪除`WordPress_App`應用程式 CodeDeploy：

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在瀏覽窗格中，展開 [部署]，然後選擇 [應用程式]。
3. 在應用程式清單中，選擇 [WordPress_App]。

4. 在 Application details (應用程式詳細資訊) 頁面上，選擇 Delete application (刪除應用程式)。
5. 當系統出現提示時，請輸入應用程式的名稱，以確認要執行刪除動作，接著選擇 Delete (刪除)。

若要刪除「系 Systems Manager 狀態管理員」關聯：

1. 請在以下位置開啟 AWS Systems Manager 主控台。https://console.aws.amazon.com/systems-manager
2. 在導覽窗格中，選擇 State Manager (狀態管理員)。
3. 選擇您建立的關聯，然後選擇 Delete (刪除)。

後續步驟？

如果您已到達這裡，恭喜您！您已成功完成 CodeDeploy部署，然後更新網站的程式碼並重新部署。

教程：部署一個「你好，世界！」 應用程序 CodeDeploy（視窗服務器）

在本教學中，您將單一網頁部署到執行網際網路資訊服務 (IIS) 做為其網頁伺服器的單一 Windows 伺服器 Amazon EC2 執行個體。這個網頁顯示一個簡單的「你好，世界！」 消息。

不是您想找的內容嗎？

- 若要練習部署到 Amazon Linux 或 RHEL (RHEL) Amazon EC2 執行個體，請參閱教學課程：部署 WordPress 至 Amazon EC2 執行個體 (Amazon Linux 或紅帽企業 Linux、macOS 或 Unix)。
- 若要練習改為部署至現場部署執行個體，請參閱教學課程：使用 CodeDeploy (Windows 伺服器、Ubuntu 伺服器或 RHEL) 將應用程式部署到內部部署執行個體。

本教學課程的步驟會從 Windows 觀點進行說明。雖然您可以在執行 Linux、macOS 或 Unix 的本機電腦上完成上述大部分步驟，但您必須調整涵蓋 Windows 型目錄路徑 (例如`c:\temp`. 此外，如果您想要連接到 Amazon EC2 執行個體，則需要一個可以透過遠端桌面通訊協定 (RDP) 連接到執行 Windows 伺服器的 Amazon EC2 執行個體的用戶端應用程式。(根據預設，Windows 包含 RDP 連線用戶端應用程式)。

開始本教學課程之前，您必須完成中的先決條件開始使用 CodeDeploy，包括設定使用者、安裝或升級 AWS CLI，以及建立 IAM 執行個體設定檔和服務角色。

主題

步驟 1：啟動視窗伺服器 Amazon EC2 執行個體

若要使用部署 Hello World 應用程式 CodeDeploy，您需要一個執行 Windows 伺服器的 Amazon EC2 執行個體。

請遵循中的說明進行創建一個 Amazon EC2 實例 CodeDeploy 當您準備好將 Amazon EC2 執行個體標籤指派給執行個體時，請務必指定的標籤金鑰**Name**和標籤值**CodeDeployDemo**。(如果您指定不同的標籤鍵或標籤值，步驟 4：部署您的 Hello World 應用程式中的說明可能會產生非預期的結果)。

啟動 Amazon EC2 執行個體之後，請返回此頁面，並繼續下一節。請不要繼續進行建立應用程式 CodeDeploy做為下一步。

Connect 到您的 Amazon EC2 實例

啟動 Amazon EC2 執行個體之後，請按照下列指示練習連線到該執行個體。

> **Note**
>
> 在這些說明中，假設您正在執行 Windows 和 Windows 桌面連線用戶端應用程式。如需詳細資訊，請參閱使用 RDP 連線至 Windows 執行個體。您可能需要針對其他作業系統或其他 RDP 連線用戶端應用程式調整這些說明。

1. 登入 AWS Management Console 並開啟 Amazon EC2 主控台，網址為 https://console.aws.amazon.com/ec2/。
2. 在導覽窗格的 Instances (執行個體) 下方，選擇 Instances (執行個體)。
3. 在清單中瀏覽並選擇您的 Windows 伺服器執行個體。
4. 選擇連線。
5. 選擇 Get Password (取得密碼)，然後選擇 Choose File (選擇檔案)。

6. 瀏覽並選擇與 Windows 伺服器亞馬遜 EC2 執行個體相關聯的 Amazon EC2 執行個體 key pair 檔案，然後選擇 [開啟]。
7. 選擇 Decrypt Password (解密密碼)。請記下顯示的密碼。您在步驟 10 會用到。
8. 選擇 Download Remote Desktop File (下載遠端桌面檔案)，然後開啟檔案。
9. 即使無法識別遠端連線發佈者的情況下，如果系統依然提示您連線，請繼續。
10. 輸入您於步驟 7 記下的密碼，然後繼續。(如果您的 RDP 連線用戶端應用程式提示您輸入使用者名稱，請輸入 **Administrator**)。
11. 在即使無法驗證遠端電腦身分的情況下，如果系統依然提示您連線，請繼續。
12. 連線之後，會顯示執行 Windows 伺服器之 Amazon EC2 執行個體的桌面。
13. 您現在可以中斷與 Amazon EC2 執行個體的連線。

> **Warning**
>
> 請不要停止或終止執行個體。否則， CodeDeploy 無法部署到它。

新增允許 HTTP 流量傳入您的視窗伺服器 Amazon EC2 執行個體的輸入規則

下一步會確認您的 Amazon EC2 執行個體具有開放的 HTTP 連接埠，因此您可以在瀏覽器中查看已部署的網頁。

1. 登入 AWS Management Console 並開啟 Amazon EC2 主控台，網址為 https://console.aws.amazon.com/ec2/。
2. 選擇執行個體，然後選擇您的執行個體。
3. 在 [說明] 索引標籤的 [安全性群組] 下，選擇 [檢視輸入規則]

   您應該會看到安全性群組中的規則清單，如下所示：

   ```
   Security Groups associated with i-1234567890abcdef0
    Ports     Protocol     Source     launch-wizard-N
    22        tcp          0.0.0.0/0          #
   ```

4. 在安全群組下，選擇 Amazon EC2 執行個體的安全群組。其可能被命名為 **launch-wizard-*N***。名稱中的 ***N*** 是一個您的執行個體建立時指派到安全群組的數字。

   選擇 Inbound (傳入) 標籤。如果執行個體的安全性群組設定正確，您應該會看到具有下列值的規則：

- 類型：HTTP
- Protocol (通訊協定)：TCP
- Port Range (連接埠範圍)：80
- 資料來源:0.0.0/0

5. 如果您沒有看到具有這些值的規則，請使用將規則新增至安全性群組中的程序，將其新增至新的安全性規則。

步驟 2：將您的來源內容設定為部署到視窗伺服器亞馬遜 EC2 執行個體

現在是時候設定應用程式的來源內容了，讓您擁有可以部署到 Amazon EC2 執行個體的項目。在本教學課程中，您會將單一網頁部署到執行 Windows 伺服器的 Amazon EC2 執行個體，該執行個體會將網際網路資訊服務 (IIS) 做為其網頁伺服器執行。這個網頁將顯示一個簡單的「你好，世界！」 消息。

主題

- 建立網頁
- 建立執行應用程式的指令碼
- 新增應用程式規格檔案

建立網頁

1. 在 `c:\temp` 資料夾中建立名為 `HelloWorldApp` 的子目錄 (子資料夾)，然後切換至該資料夾。

```
mkdir c:\temp\HelloWorldApp
cd c:\temp\HelloWorldApp
```

> **Note**
>
> 您不需要使用 `c:\temp` 位置或 `HelloWorldApp` 子資料夾名稱。如果您使用不同的位置或子資料夾名稱，請務必在本教學中予以使用。

2. 使用文字編輯器，在資料夾內建立檔案。將檔案命名為 `index.html`。

```
notepad index.html
```

3. 將下列 HTML 程式碼新增至檔案，然後儲存檔案。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/
TR/html4/loose.dtd">
<html>
<head>
  <title>Hello, World!</title>
  <style>
    body {
      color: #ffffff;
      background-color: #0188cc;
      font-family: Arial, sans-serif;
      font-size:14px;
    }
  </style>
</head>
<body>
  <div align="center"><h1>Hello, World!</h1></div>
  <div align="center"><h2>You have successfully deployed an application using
 CodeDeploy</h2></div>
  <div align="center">
    <p>What to do next? Take a look through the <a href="https://aws.amazon.com/
codedeploy">CodeDeploy Documentation</a>.</p>
  </div>
</body>
</html>
```

建立執行應用程式的指令碼

接下來，您將建立一個 CodeDeploy 指令碼，用於在目標 Amazon EC2 執行個體上設定 Web 伺服器。

1. 在 index.html 檔案儲存所在的相同子資料夾中，使用文字編輯器來建立另一個檔案。將檔案命名為 before-install.bat。

```
notepad before-install.bat
```

2. 將下列批次指令碼程式碼新增至檔案，然後儲存檔案。

```
REM Install Internet Information Server (IIS).
c:\Windows\Sysnative\WindowsPowerShell\v1.0\powershell.exe -Command Import-Module -
Name ServerManager
```

```
c:\Windows\Sysnative\WindowsPowerShell\v1.0\powershell.exe -Command Install-
WindowsFeature Web-Server
```

新增應用程式規格檔案

接下來，除了網頁和批處理腳本文件之外，您還將添加一個應用程序規格文件（AppSpec 文件）。此 AppSpec 檔案是 YAML 格式的檔案，可用來 CodeDeploy 執行下列作業：

- 將應用程式修訂中的來源檔案，映射至執行個體上的目標。
- 指定要在部署期間於執行個體上執行的指令碼。

檔 AppSpec 案必須命名`appspec.yml`。它必須放置在應用程式來源碼的根資料夾中。

1. 在 `index.html` 和 `before-install.bat` 檔案儲存所在的相同子資料夾中，使用文字編輯器來建立另一個檔案。將檔案命名為 `appspec.yml`。

   ```
   notepad appspec.yml
   ```

2. 將下列 YAML 程式碼新增至檔案，然後儲存檔案。

   ```
   version: 0.0
   os: windows
   files:
     - source: \index.html
       destination: c:\inetpub\wwwroot
   hooks:
     BeforeInstall:
       - location: \before-install.bat
         timeout: 900
   ```

CodeDeploy 將使用此 AppSpec 檔案將應用程式原始程式碼根資料夾中的`index.html`檔案複製到目標 Amazon EC2 執行個體上的`c:\inetpub\wwwroot`資料夾。在部署期間， CodeDeploy 將在部**`BeforeInstall`**署生命週期事件期間在目標 Amazon EC2 執行個體上執行`before-install.bat`批次指令碼。如果此指令碼執行時間超過 900 秒 (15 分鐘)，則 CodeDeploy 會停止部署，並將 Amazon EC2 執行個體的部署標記為失敗。

如需這些設定的詳細資訊，請參閱CodeDeploy AppSpec 檔案參考。

> **Important**
>
> 此檔案中每個項目之間的空格位置和數目十分重要。如果間距不正確， CodeDeploy 將引發可能難以調試的錯誤。如需更多詳細資訊，請參閱 AppSpec 檔案間距。

第 3 步：上傳您的「你好，世界！」 應用到 Amazon S3

現在，您將準備源內容並將其上傳到 CodeDeploy 可以部署它的位置。下列指示說明如何佈建 Amazon S3 儲存貯體、準備儲存貯體的應用程式修訂版檔案、捆綁修訂版本的檔案，然後將修訂推送至儲存貯體。

> **Note**
>
> 雖然本教學課程未說明，但您可以使用 CodeDeploy 將應用程式從 GitHub 儲存庫部署到執行個體。如需詳細資訊，請參閱 CodeDeploy 與整合 GitHub。

主題

- 佈建 Amazon S3 存儲桶
- 為值區準備應用程式的檔案
- 將應用程序的文件捆綁到單個存檔文件中並推送歸檔文件

佈建 Amazon S3 存儲桶

在 Amazon S3 中建立儲存容器或儲存貯體，或使用現有的儲存貯體。確保您可以將修訂版上傳到存儲桶，並且部署中使用的 Amazon EC2 執行個體可以從存儲桶下載修訂版本。

您可以使用 AWS CLI Amazon S3 控制台或 Amazon S3 API 來創建一個 Amazon S3 存儲桶。建立值區之後，請務必授與值區和 CodeDeploy使用者的存取權限。

> **Note**
>
> 所有 AWS 帳戶的儲存貯體名稱在 Amazon S3 中都必須是唯一的。如果您無法使用 **codedeploydemobucket**，請嘗試不同的儲存貯體名稱 (例如後接破折號和您的縮寫或其他唯一識別符的 **codedeploydemobucket**)。然後，請務必將在本教學中看到的所有 **codedeploydemobucket** 都替代為您的儲存貯體名稱。

Amazon S3 儲存貯體必須在啟動目標 Amazon EC2 執行個體的相同 AWS 區域中建立。例如，如果您在美國東部 (維吉尼亞北部) 區域建立儲存貯體，則必須在美國東部 (維吉尼亞北部) 區域啟動目標 Amazon EC2 執行個體。

主題

- 若要建立 Amazon S3 儲存貯體 (CLI)
- 若要建立 Amazon S3 儲存貯體 (主控台)
- 授予 Amazon S3 儲存貯體和您的 AWS 帳戶的許可

若要建立 Amazon S3 儲存貯體 (CLI)

呼叫命mb令以建立名為的 Amazon S3 儲存貯體**codedeploydemobucket**：

```
aws s3 mb s3://codedeploydemobucket --region region
```

若要建立 Amazon S3 儲存貯體 (主控台)

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
2. 在 Amazon S3 主控台中，選擇建立儲存貯體。
3. 在 Bucket name (儲存貯體名稱) 方塊中，輸入儲存貯體的名稱。
4. 在 Region (區域) 清單中，選擇目標區域，然後選擇 Create (建立)。

授予 Amazon S3 儲存貯體和您的 AWS 帳戶的許可

您必須擁有上傳到 Amazon S3 儲存貯體的許可。您可以透過 Amazon S3 儲存貯體政策指定這些許可。例如，在下列 Amazon S3 儲存貯體政策中，使用萬用字元 (*) 可讓 AWS 帳戶111122223333將檔案上傳到 Amazon S3 儲存貯體中名為的任何目錄codedeploydemobucket：

```
{
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
            ],
            "Effect": "Allow",
```

```
            "Resource": "arn:aws:s3:::codedeploydemobucket/*",
            "Principal": {
                "AWS": [
                    "111122223333"
                ]
            }
        }
    ]
}
```

若要檢視您的 AWS 帳戶 ID，請參閱尋找您的 AWS 帳戶 ID。

現在是驗證 Amazon S3 儲存貯體允許從每個參與的 Amazon EC2 執行個體下載請求的好時機。您可以透過 Amazon S3 儲存貯體政策指定。例如，在下列 Amazon S3 儲存貯體政策中，使用萬用字元 (*) 可讓任何具`codedeploydemobucket`有附加 IAM 執行個體設定檔包含 ARN `arn:aws:iam::444455556666:role/CodeDeployDemo` 的 Amazon EC2 執行個體從 Amazon S3 儲存貯體中的任何目錄下載檔案：

```
{
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::codedeploydemobucket/*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666:role/CodeDeployDemo"
                ]
            }
        }
    ]
}
```

如需如何產生和連接 Amazon S3 儲存貯體政策的詳細資訊，請參閱儲存貯體政策範例。

您在其中建立 CodeDeploy 的管理員使用者也步驟 1：設定必須擁有將修訂上傳到 Amazon S3 儲存貯體的權限。指定這一點的方法之一是透過 IAM 政策，將其新增至使用者的權限集，或是 IAM 角色 (您允許使用者假設)。下列 IAM 政策允許使用者將修訂上傳到名為的 Amazon S3 儲存貯體中的任何位置`codedeploydemobucket`：

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":["s3:PutObject"],
      "Resource":"arn:aws:s3:::codedeploydemobucket/*"
    }
  ]
}
```

如需如何建立 IAM 政策的詳細資訊，請參閱 IAM 使用者指南中的建立 IAM 政策。如需有關將原則新增至權限集的資訊，請參閱《AWS IAM Identity Center 使用指南》中的「建立權限集」。

為值區準備應用程式的檔案

確保網頁， AppSpec 文件和腳本在您的開發計算機上進行組織，如下所示：

```
c:\
  |-- temp\
        |--HelloWorldApp\
             |-- appspec.yml
             |-- before-install.bat
             |-- index.html
```

將應用程序的文件捆綁到單個存檔文件中並推送歸檔文件

將檔案綁定到封存檔案 (稱為應用程式「修訂版本」)。

> **Note**
>
> 可能會向您收取下列作業的費用：在儲存貯體中存放物件，以及將應用程式修訂傳入和傳出儲存貯體。如需詳細資訊，請參閱 Simple Storage Service (Amazon S3) 定價。

1. 在開發電腦上，切換至檔案存放所在的資料夾：

   ```
   cd c:\temp\HelloWorldApp
   ```

> **Note**
>
> 如果您未切換至此資料夾，將會在目前的資料夾開始檔案綁定。例如，如果您的目前資料夾是 `c:\temp`，而不是 `c:\temp\HelloWorldApp`，將會開始綁定 `c:\temp` 資料夾中的檔案和子資料夾，而此資料夾可能不只包含 `HelloWorldApp` 子資料夾。

2. 呼叫指create-application令來註冊名為下列名稱**`HelloWorld_App`**的新應用程式 CodeDeploy：

```
aws deploy create-application --application-name HelloWorld_App
```

3. 呼叫 CodeDeploy push 命令將檔案捆綁在一起、將修訂上傳到 Amazon S3，然後在一個動作中註冊有 CodeDeploy 關上傳修訂的資訊。

```
aws deploy push --application-name HelloWorld_App --s3-location s3://
codedeploydemobucket/HelloWorld_App.zip --ignore-hidden-files
```

此指令會將目前目錄中的檔案 (不包括任何隱藏檔案) 捆綁到一個名為的單一封存檔案中`HelloWorld_App.zip`，將修訂上載到**`codedeploydemobucket`**值區，並將有 CodeDeploy 關上載修訂的資訊註冊到值區。

步驟 4：部署您的 Hello World 應用程式

現在，您可以部署您上傳到 Amazon S3 的 Hello World 應用程式修訂版範例。您可以使用 AWS CLI 或主 CodeDeploy 控台部署修訂版本並監視部署進度。成功部署應用程式修訂版之後，您要檢查結果。

主題

- 部署您的應用程式修訂 CodeDeploy
- 監控和疑難排解您的部署
- 驗證您的部署

部署您的應用程式修訂 CodeDeploy

您可以使用 CLI 或主控台部署您的應用程式。

主題

- 部署應用程式修訂 (CLI)
- 部署應用程式修訂 (主控台)

部署應用程式修訂 (CLI)

1. 首先，部署需要部署群組。不過，在您建立部署群組之前，需要服務角色 ARN。服務角色是一種 IAM 角色，可提供代表您行動的服務權限。在這種情況下，服務角色 CodeDeploy 授予存取 Amazon EC2 執行個體的權限，以擴展 (讀取) 其 Amazon EC2 執行個體標籤。

   您應該已經遵循建立服務角色 (CLI) 中的說明，來建立服務角色。若要取得服務角色的 ARN，請參閱取得服務角色 ARN (CLI)。
2. 現在您已經擁有 ARN，請使用具有服務角色 ARN 的 Amazon EC2 create-deployment-group 執行個體標籤**CodeDeployDemo**和名為**HelloWorld_DepGroup**部署組態的 Amazon EC2 執行個體標籤來建立名**HelloWorld_App**為的部署群組**CodeDeployDefault.OneAtATime**，以建立與名為的應用程式相關聯的部署群組：

   ```
   aws deploy create-deployment-group --application-name HelloWorld_App
    --deployment-group-name HelloWorld_DepGroup --deployment-
   config-name CodeDeployDefault.OneAtATime --ec2-tag-filters
    Key=Name,Value=CodeDeployDemo,Type=KEY_AND_VALUE --service-role-arn serviceRoleARN
   ```

   > **Note**
   >
   > 該create-deployment-group命令支援建立觸發器，以便將 Amazon SNS 通知傳送給主題訂閱者有關部署和執行個體中指定事件的主題訂閱者。該命令還支援自動復原部署和設定警示以停止部署的選項，以便在符合 Amazon CloudWatch 警示中的監控閾值時停止部署。這些動作的指令不包括在本教學課程中。
3. 建立部署之前，部署群組中的執行個體必須已安裝 CodeDeploy 代理程式。您可以使用 AWS Systems Manager 與下列命令，從命令列安裝代理程式：

   ```
   aws ssm create-association --name AWS-ConfigureAWSPackage
    --targets Key=tag:Name,Values=CodeDeployDemo --parameters
    action=Install,name=AWSCodeDeployAgent --schedule-expression "cron(0 2 ? * SUN
    *)"
   ```

   此命令會在系統管理員狀態管理員中建立一個關聯，以安裝 CodeDeploy 代理程式，然後嘗試在每個星期天早上 2:00 進行更新。如需 CodeDeploy 代理程式的詳細資訊，請參閱使用

CodeDeploy 代理程式。如需有關「Systems Manager 的詳細資訊，請參閱何謂 AWS Systems Manager。

4. 現在，請在名為 **codedeploydemobucket** 的儲存貯體中使用名為 **HelloWorld_App.zip** 的應用程式修訂，呼叫 create-deployment 命令來建立與名為 **HelloWorld_App** 之應用程式、名為 **CodeDeployDefault.OneAtATime** 之部署組態和名為 **HelloWorld_DepGroup** 之部署群組建立關聯的部署：

```
aws deploy create-deployment --application-name HelloWorld_App --deployment-config-name CodeDeployDefault.OneAtATime --deployment-group-name HelloWorld_DepGroup --s3-location bucket=codedeploydemobucket,bundleType=zip,key=HelloWorld_App.zip
```

部署應用程式修訂 (主控台)

1. 使用 CodeDeploy 主控台部署應用程式修訂版本之前，您需要服務角色 ARN。服務角色是一種 IAM 角色，可提供代表您行動的服務權限。在這種情況下，服務角色 CodeDeploy 授予存取 Amazon EC2 執行個體的權限，以擴展 (讀取) 其 Amazon EC2 執行個體標籤。

   您應該已經遵循建立服務角色 (主控台) 中的說明，來建立服務角色。若要取得服務角色的 ARN，請參閱取得服務角色 ARN (主控台)。
2. 現在您已經擁有 ARN，您可以使用 CodeDeploy 主控台來部署應用程式修訂版本。

   請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

3. 在瀏覽窗格中，展開 [部署]，然後選擇 [應用程式]。
4. 選擇 [HelloWorld應用程式]。
5. 在 Deployment groups (部署群組) 標籤中，選擇 Create deployment group (建立部署群組)。
6. 在 Deployment group name (部署群組名稱) 中，輸入 **HelloWorld_DepGroup**。
7. 在 Service Role (服務角色) 中，選擇服務角色的名稱。
8. 在 Deployment type (部署類型) 中，選擇 In-place (就地)。
9. 在環境組態中，選取 Amazon EC2 執行個體。
10. 在 [代理程式組態使用] 中 AWS Systems Manager，保留預設值。

11. 在 Key (金鑰) 中，輸入 **Name**。
12. 在 Value (值) 中輸入 **CodeDeployDemo**。
13. 在 [部署組態] 中，選擇CodeDeployDefault。 OneAtA 時間。
14. 在 Load Balancer (負載平衡器) 中，清除 Enable load balancing (啟用負載平衡)。
15. 選擇 Create deployment group (建立部署群組)。
16. 選擇 Create deployment (建立部署)。
17. 在部署組中，選擇 HelloWorld_ DepGroup
18. 在修訂類型中，選擇我的應用程式存放在 Amazon S3，然後在修訂位置中，輸入先前上傳到 Amazon S3 的 Hello World 應用程式修訂範例位置。取得位置：

    a. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。

    b. 在值區清單中，選擇 codedeploydemobucket (或您上傳應用程式修訂版的值區名稱)。

    c. 在物件清單中，選擇 HelloWorld_App.zip。

    d. 在 Overview (概觀) 標籤上，選擇 Copy path (複製路徑)。

    e. 返回 CodeDeploy 主控台，然後在「修訂位置」中貼上「連結」欄位值。
19. 針對 Revision file type (修訂檔案類型)，選擇 .zip。
20. (選用) 在 Deployment description (部署描述) 中輸入註解。
21. 選擇 Create deployment (建立部署)。新建立部署的相關資訊會顯示在 Deployments (部署) 頁面上。

監控和疑難排解您的部署

使用 AWS CLI 或主控台來監控部署並進行疑難排解。

主題

- 監控部署並進行疑難排解 (CLI)
- 監控和故障診斷部署 (主控台)

監控部署並進行疑難排解 (CLI)

1. 針對名為 **HelloWorld_App** 的應用程式和名為 **HelloWorld_DepGroup** 的部署群組呼叫 list-deployments 命令，來取得部署 ID：

```
aws deploy list-deployments --application-name HelloWorld_App --deployment-group-
name HelloWorld_DepGroup --query "deployments" --output text
```

2. 使用部署 ID，呼叫 get-deployment 命令：

```
aws deploy get-deployment --deployment-id deploymentID --query
 "deploymentInfo.status" --output text
```

3. 此命令傳回部署的整體狀態。如果成功，值為 Succeeded。

   如果整體狀態為Failed，您可以呼叫list-deployment-instances和之類的命令get-deployment-instance來進行疑難排解。如需其他故障診斷選項，請參閱分析日誌檔案以調查執行個體的部署失敗。

監控和故障診斷部署 (主控台)

在 CodeDeploy 主控台的「部署」頁面上，您可以在「狀態」欄中監視部署的狀態。

若要取得部署的詳細資訊，特別是在 Status (狀態) 欄位中，有任何數值不是 Succeeded (成功) 的情況下，則可執行以下動作：

1. 在 Deployments (部署) 資料表中，選擇部署 ID。在部署失敗之後，說明失敗原因的訊息會顯示在部署的詳細資訊頁面中。
2. 。 此時會顯示有關部署執行個體的詳細資訊。部署失敗後，您可能可以判斷哪些 Amazon EC2 執行個體以及部署失敗的步驟。
3. 您可以使用View Instance Details中所述的這類技術，藉此執行其他疑難排解。您也可以分析 Amazon EC2 執行個體上的部署日誌檔。如需詳細資訊，請參閱 分析日誌檔案以調查執行個體的部署失敗。

驗證您的部署

在您的部署成功之後，請驗證安裝運作中。使用 Amazon EC2 執行個體的公有 DNS 地址在網頁瀏覽器中檢視網頁。若要取得公有 DNS 值，請在 Amazon EC2 主控台中選擇 Amazon EC2 執行個體，然後在「描述」索引標籤上尋找公用 DNS 中的值。)

例如，如果您的 Amazon EC2 執行個體的公有 DNS 地址是**ec2-01-234-567-890.compute-1.amazonaws.com**，您可以使用下列 URL：

```
http://ec2-01-234-567-890.compute-1.amazonaws.com
```

如果成功，您應該會看到 Hello World 網頁。

第 5 步：更新和重新部署您的「你好，世界！」 應用程式

現在您已成功部署應用程式修訂版，請在開發電腦上更新網頁的程式碼，然後使 CodeDeploy 用重新部署網站。重新部署之後，您應該能夠在 Amazon EC2 執行個體上看到變更。

主題

- 修改網頁
- 重新部署網站

修改網頁

1. 移至 `c:\temp\HelloWorldApp` 子資料夾，然後使用文字編輯器修改 `index.html` 檔案：

```
cd c:\temp\HelloWorldApp
notepad index.html
```

2. 修訂 `index.html` 檔案的內容，變更網頁的背景顏色和一些文字，然後儲存檔案。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/
TR/html4/loose.dtd">
<html>
<head>
  <title>Hello Again, World!</title>
  <style>
    body {
      color: #ffffff;
      background-color: #66cc00;
      font-family: Arial, sans-serif;
      font-size:14px;
    }
  </style>
</head>
<body>
  <div align="center"><h1>Hello Again, World!</h1></div>
  <div align="center"><h2>You have successfully deployed a revision of an
 application using CodeDeploy</h2></div>
```

```
  <div align="center">
    <p>What to do next? Take a look through the <a href="https://aws.amazon.com/
codedeploy">CodeDeploy Documentation</a>.</p>
  </div>
</body>
</html>
```

重新部署網站

現在您已經修改了代碼，請使用 Amazon S3 並 CodeDeploy 重新部署網頁。

依照中所述將變更捆綁並上傳到 Amazon S3 將應用程序的文件捆綁到單個存檔文件中並推送歸檔文件。(當您遵循這些說明時，不需要建立新的應用程式)。如以前一樣將相同的金鑰給予修訂 (**HelloWorld_App.zip**)。將其上傳到您之前建立的相同 Amazon S3 儲存貯體 (例如**codedeploydemobucket**)。

使用 AWS CLI 或 CodeDeploy 主控台重新部署網站。

主題

- 重新部署網站 (CLI)
- 重新部署網站 (主控台)

重新部署網站 (CLI)

再次於名為 **codedeploydemobucket** 的儲存貯體中使用名為 **HelloWorld_App** 的應用程式、名為 **CodeDeployDefault.OneAtATime** 的部署組態、名為 **HelloWorld_DepGroup** 的部署群組和名為 **HelloWorld_App.zip** 的修訂，根據上傳的修訂以呼叫 create-deployment 命令來建立部署：

```
 aws deploy create-deployment --application-name HelloWorld_App --deployment-config-
name CodeDeployDefault.OneAtATime --deployment-group-name HelloWorld_DepGroup --s3-
location bucket=codedeploydemobucket,bundleType=zip,key=HelloWorld_App.zip
```

您可以檢查新部署的狀態，如監控和疑難排解您的部署中所述。

重新部署網站後，請在 Web 瀏覽器中重新造訪該網站，以確認網頁上的背景顏色和文字是否已變更。CodeDeploy (您可能需要重新整理瀏覽器)。如果背景顏色和文字已變更，恭喜您！您已修改並重新部署該網站！

重新部署網站 (主控台)

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在導覽窗格上，選擇 Applications (應用程式)。
3. 在 [應用程式] 清單中，選擇 [HelloWorld_App]。
4. 在 Deployments (部署) 標籤中，選擇 Create deployment (建立部署)。

   a. 在 [部署群組] 清單中，選擇 [HelloWorld_] DepGroup。

   b. 在修訂位置中，輸入修訂版本的 Amazon S3 連結。

      尋找連結值：

      i. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。

         在 Amazon S3 主控台中瀏覽並開啟程式碼部署模組**`HelloWorld_App.zip`**，然後選擇您的修訂。

      ii. 如果在 Amazon S3 主控台中看不到「屬性」窗格，請選擇「屬性」按鈕。

      iii. 在 Properties (屬性) 窗格中，複製 Link (連結) 欄位的值。

      iv. 返回 CodeDeploy 控制台，然後將鏈接粘貼到「修訂」位置。

      v.

   c. 在 Revision file type (修訂檔案類型) 中，如果顯示的訊息指出偵測不到檔案類型，請選擇 .zip。

   d. 將 Deployment description (部署描述) 空白。

   e. 展開部署群組覆寫在 [部署規劃] 清單中，選擇CodeDeployDefault。 OneAtaTime，然後選擇 [建立部署]。

      接著，您便能檢查部署的狀態，如監控和疑難排解您的部署中所述。

重新部署網站後，請在 Web 瀏覽器中重新造訪該網站，以確認網頁上的背景顏色和文字是否已變更。 CodeDeploy (您可能需要重新整理瀏覽器)。如果背景顏色和文字已變更，恭喜您！您已修改並重新部署該網站！

第 6 步：清理你的「你好，世界！」 應用程式及相關資源

您現在已成功更新到「你好，世界！」 編碼並重新部署網站。為了避免完成本教學課程所建立的資源持續發生費用，您應該刪除：

- 任何 AWS CloudFormation 堆疊 (或終止任何 Amazon EC2 執行個體，如果您在以外建立它們 AWS CloudFormation)。
- 任何 Amazon S3 桶。
- CodeDeploy 中的 `HelloWorld_App` 應用程式。
- CodeDeploy 代理程式的 AWS Systems Manager 狀態管理員關聯。

您可以使用 AWS CLI、 AWS CloudFormation、Amazon S3、Amazon EC2 和 CodeDeploy 主控台或 AWS API 來執行清理。

主題

若要使用清除資源(CLI)

1. 如果您在本教學課程中使用了 AWS CloudFormation 堆疊，請針對名為的堆疊呼叫delete-stack命令來刪除堆疊**CodeDeployDemoStack**。這會終止所有隨附的 Amazon EC2 執行個體，並刪除堆疊最初建立的所有隨附 IAM 角色。

   ```
   aws cloudformation delete-stack --stack-name CodeDeployDemoStack
   ```

2. 若要刪除 Amazon S3 儲存貯體，請對名為的儲存貯體使用--recursive交換器呼叫rm命令**codedeploydemobucket**。這會刪除儲存貯體以及儲存貯體中的所有物件。

   ```
   aws s3 rm s3://codedeploydemobucket --recursive --region region
   ```

3. 若要從中刪除HelloWorld_App應用程式 CodeDeploy，請呼叫delete-application指令。這會刪除應用程式的所有相關聯部署群組記錄和部署記錄。

```
aws deploy delete-application --application-name HelloWorld_App
```

4. 若要刪除「Systems Manager 狀態管理員」關聯，請呼叫delete-association指令。

```
aws ssm delete-association --assocation-id association-id
```

您可以通過調用命###### ID。describe-association

```
aws ssm describe-association --name AWS-ConfigureAWSPackage --targets
 Key=tag:Name,Values=CodeDeployDemo
```

5. 如果您在本教學中未使用 AWS CloudFormation 堆疊，請呼叫terminate-instances命令以終止您手動建立的 Amazon EC2 執行個體。提供要終止之 Amazon EC2 執行個體的 ID。

```
aws ec2 terminate-instances --instance-ids instanceId
```

清除資源 (主控台)

如果您在本教程中使用了我們的 AWS CloudFormation 模板，請刪除關聯的 AWS CloudFormation 堆棧。

1. 請登入 AWS Management Console 並開啟 AWS CloudFormation 主控台，網址為 https://console.aws.amazon.com/cloudformation。
2. 在搜尋方塊中，輸入 AWS CloudFormation 堆疊名稱 (例如，**CodeDeployDemoStack**)。
3. 選取堆疊名稱旁的方塊。
4. 在 Actions (動作) 選單中，選擇 Delete Stack (刪除堆疊)。這會刪除堆疊、終止所有隨附的 Amazon EC2 執行個體，以及刪除所有隨附的 IAM 角色。

若要終止您在 AWS CloudFormation 堆疊外部建立的 Amazon EC2 執行個體：

1. 登入 AWS Management Console 並開啟 Amazon EC2 主控台，網址為 https://console.aws.amazon.com/ec2/。
2. 在 Instances (執行個體) 區域中，選擇 Instances (執行個體)。
3. 在搜尋方塊中，輸入您要終止的 Amazon EC2 執行個體名稱，然後按 Enter 鍵。

4. 選擇 Amazon EC2 實例。
5. 選擇 Actions (動作)，指向 Instance State (執行個體狀態)，然後選擇 Terminate (終止)。出現提示時，選擇 Yes, Terminate (是，終止)。對任何其他 Amazon EC2 執行個體重複這些步驟。

要刪除 Amazon S3 存儲桶：

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在儲存貯體清單中，瀏覽並選擇 Amazon S3 儲存貯體的名稱 (例如**codedeploydemobucket**)。
3. 您必須先刪除其內容，才能刪除儲存貯體。選擇儲存貯體中的所有檔案，例如 **HelloWorld_App.zip**。在操作功能表中，選擇刪除。出現提示要您確認刪除時，選擇 OK (確定)。
4. 儲存貯體清空之後，您即可刪除儲存貯體。在儲存貯體清單中，選擇儲存貯體的資料列 (但不是儲存貯體名稱)。選擇 Delete bucket (刪除儲存貯體)，然後在出現確認提示時，選擇 OK (確定)。

若要從中刪除`HelloWorld_App`應用程式 CodeDeploy：

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在瀏覽窗格中，展開 [部署]，然後選擇 [應用程式]。
3. 選擇 **HelloWorld_App**。
4. 選擇刪除應用程式。
5. 當出現提示時，輸入 **Delete**，然後選擇 Delete (刪除)。

若要刪除「系 Systems Manager 狀態管理員」關聯：

1. 請在以下位置開啟 AWS Systems Manager 主控台。https://console.aws.amazon.com/systems-manager
2. 在導覽窗格中，選擇 State Manager (狀態管理員)。

3. 選擇您建立的關聯，然後選擇 Delete (刪除)。

後續步驟？

如果您已經到達這裡，您已經成功完成了 CodeDeploy. 恭喜您！

教學課程：使用 CodeDeploy (Windows 伺服器、Ubuntu 伺服器或 RHEL) 將應用程式部署到內部部署執行個體

本教學課程可指導您將範例應用程式修訂部署到單一現場部署執行個體 (也就是執行 Windows 伺服器、Ubuntu 伺服器或 RHEL) 並非 Amazon EC2 執行個體的實體裝置， CodeDeploy 藉此協助您獲得使用經驗。如需內部部署執行個體及其使用方式的相關資訊 CodeDeploy，請參閱Working with On-Premises Instances。

不是您想找的內容嗎？

- 若要練習部署到執行 Amazon Linux 或 RHEL 的亞馬遜 EC2 執行個體，請參閱教學課程：部署 WordPress 至 Amazon EC2 執行個體 (Amazon Linux 或紅帽企業 Linux、macOS 或 Unix)。
- 若要練習部署到執行 Windows 伺服器的 Amazon EC2 執行個體，請參閱教程：部署一個「你好，世界！」 應用程序 CodeDeploy（視窗服務器）。

主題

必要條件

開始此教學課程之前，您必須先完成中的先決條件開始使用 CodeDeploy，其中包括設定使用者、安裝或升級 AWS CLI，以及建立服務角色。您不必按照先決條件中所述建立 IAM 執行個體設定檔。內部部署執行個體不使用 IAM 執行個體設定檔

您將設定為現場部署執行個體的實體裝置，必須執行 CodeDeploy 代理程式支援的作業系統 中所列的其中一個作業系統。

步驟 1：設定內部部署執行個體

您必須先進行設定，之後才能部署到現場部署執行個體。請遵循Working with On-Premises Instances中的指示，然後返回此頁面。

安裝 CodeDeploy 代理程式

設定內部部署執行個體後，請依照安裝 CodeDeploy 代理程式中針對內部部署執行個體的步驟執行，並返回此頁面。

步驟 2：建立範例應用程式修訂

在此步驟中，您將建立範例應用程式修訂版，以部署到您的內部部署執行個體。

由於很難知道在內部部署執行個體上已安裝哪些軟體和功能，或者組織的原則允許安裝哪些軟體和功能，因此我們在此提供的範例應用程式修訂只是使用批次指令碼 (適用於 Windows Server) 或殼層指令碼 (適用於 Ubuntu Server 和 RHEL)，將文字檔寫入內部部署執行個體上的位置。每個 CodeDeploy 部署生命週期事件都會寫入一個檔案，包括 Install AfterInstall、ApplicationStart、、和ValidateService。在BeforeInstall部署生命週期事件期間，會執行指令碼以移除先前部署此範例期間撰寫的舊檔案，並在內部部署執行個體上建立要寫入新檔案的位置。

> **Note**
>
> 這個範例應用程式修訂版可能無法進行部署，若以下任何一項為真：
>
> - 在內部部署執行個體上啟動 CodeDeploy 代理程式的使用者沒有執行指令碼的權限。
> - 使用者沒有在指令碼中所列位置建立或刪除資料夾的權限。
> - 使用者沒有在指令碼中列出的位置建立文字檔的權限。

> **Note**
>
> 如果您已設定 Windows Server 執行個體，並且想要部署不同的範例，您可能需要使用教程：部署一個「你好，世界！」 應用程序 CodeDeploy（視窗服務器）教學課程步驟 2：將您的來源內容設定為部署到視窗伺服器亞馬遜 EC2 執行個體中的範例。

如果您已設定 RHEL 執行個體，並想要部署不同的範例，您可能需要使用教學課程：部署 WordPress 至 Amazon EC2 執行個體 (Amazon Linux 或紅帽企業 Linux、macOS 或 Unix)教學課程步驟 2：設定要部署到 Amazon Linux 或 RHEL 亞馬遜 EC2 執行個體的來源內容中的範例。
目前，Ubuntu 服務器沒有替代示例。

1. 請在您的開發機器上，建立一個名為 CodeDeployDemo-OnPrem 的子目錄 (子資料夾)，將儲存範例應用程式修訂版的檔案，然後切換到子資料夾。在此範例中，我們假設您將使用c:\temp資料夾做為 Windows 伺服器的根資料夾，或使用/tmp資料夾做為 Ubuntu 伺服器和 RHEL 的根資料夾。如果您使用不同資料夾，請務必在此教學課程中都替換為這個。

   針對 Windows：

   ```
   mkdir c:\temp\CodeDeployDemo-OnPrem
   cd c:\temp\CodeDeployDemo-OnPrem
   ```

   若為 Linux、macOS 或 Unix：

   ```
   mkdir /tmp/CodeDeployDemo-OnPrem
   cd /tmp/CodeDeployDemo-OnPrem
   ```

2. 在 CodeDeployDemo-OnPrem 子資料夾的根目錄中，使用純文字編輯器建立兩個分別名為 appspec.yml 和 install.txt 的檔案：

   appspec.yml適用於視窗伺服器：

   ```
   version: 0.0
   os: windows
   files:
     - source: .\install.txt
       destination: c:\temp\CodeDeployExample
   hooks:
     BeforeInstall:
       - location: .\scripts\before-install.bat
         timeout: 900
     AfterInstall:
       - location: .\scripts\after-install.bat
         timeout: 900
     ApplicationStart:
   ```

```
    - location: .\scripts\application-start.bat
      timeout: 900
  ValidateService:
    - location: .\scripts\validate-service.bat
      timeout: 900
```

appspec.yml對於 Ubuntu 服務器和 RHEL：

```
version: 0.0
os: linux
files:
  - source: ./install.txt
    destination: /tmp/CodeDeployExample
hooks:
  BeforeInstall:
    - location: ./scripts/before-install.sh
      timeout: 900
  AfterInstall:
    - location: ./scripts/after-install.sh
      timeout: 900
  ApplicationStart:
    - location: ./scripts/application-start.sh
      timeout: 900
  ValidateService:
    - location: ./scripts/validate-service.sh
      timeout: 900
```

若要取得有關 AppSpec 檔案的更多資訊，請參閱將應用程式規格檔案新增至修訂 CodeDeploy和CodeDeploy AppSpec 檔案參考。

install.txt:

```
The Install deployment lifecycle event successfully completed.
```

3. 在 CodeDeployDemo-OnPrem 子資料夾的根目錄下，建立 scripts 子資料夾，然後切換到該資料夾：

   針對 Windows：

```
mkdir c:\temp\CodeDeployDemo-OnPrem\scripts
cd c:\temp\CodeDeployDemo-OnPrem\scripts
```

若為 Linux、macOS 或 Unix：

```
mkdir -p /tmp/CodeDeployDemo-OnPrem/scripts
cd /tmp/CodeDeployDemo-OnPrem/scripts
```

4. 在scripts子資料夾的根目錄中，validate-service.bat針對 Ubuntu 伺服器和 RHEL before-install.bat after-install.batapplication-start.bat，使用文字編輯器建立四個名validate-service.sh為before-install.shafter-install.shapplication-start.sh、、和的檔案：

   對於視窗伺服器：

   before-install.bat:

   ```
   set FOLDER=%HOMEDRIVE%\temp\CodeDeployExample

   if exist %FOLDER% (
     rd /s /q "%FOLDER%"
   )

   mkdir %FOLDER%
   ```

   after-install.bat:

   ```
   cd %HOMEDRIVE%\temp\CodeDeployExample

   echo The AfterInstall deployment lifecycle event successfully completed. > after-
   install.txt
   ```

   application-start.bat:

   ```
   cd %HOMEDRIVE%\temp\CodeDeployExample

   echo The ApplicationStart deployment lifecycle event successfully completed. >
    application-start.txt
   ```

   validate-service.bat:

   ```
   cd %HOMEDRIVE%\temp\CodeDeployExample
   ```

```
echo The ValidateService deployment lifecycle event successfully completed. >
 validate-service.txt
```

對於 Ubuntu 服務器和 RHEL：

before-install.sh:

```
#!/bin/bash
export FOLDER=/tmp/CodeDeployExample

if [ -d $FOLDER ]
then
 rm -rf $FOLDER
fi

mkdir -p $FOLDER
```

after-install.sh:

```
#!/bin/bash
cd /tmp/CodeDeployExample

echo "The AfterInstall deployment lifecycle event successfully completed." > after-
install.txt
```

application-start.sh:

```
#!/bin/bash
cd /tmp/CodeDeployExample

echo "The ApplicationStart deployment lifecycle event successfully completed." >
 application-start.txt
```

validate-service.sh:

```
#!/bin/bash
cd /tmp/CodeDeployExample

echo "The ValidateService deployment lifecycle event successfully completed." >
 validate-service.txt
```

```
unset FOLDER
```

5. 僅適用於 Ubuntu 伺服器和 RHEL，請確定四個殼層指令碼具有執行權限：

```
chmod +x ./scripts/*
```

步驟 3：將您的應用程式修訂版捆綁並上傳到 Amazon S3

在部署應用程式修訂版之前，您需要將檔案捆綁在一起，然後將檔案服務包上傳到 Amazon S3 儲存貯體。請遵循建立應用程式 CodeDeploy和將修訂推送 CodeDeploy 至 Amazon S3 (僅適用於 EC2 /內部部署)中的說明進行。(雖然您可以給予應用程式和部署群組任何名稱，我們建議您對於應用程式名稱使用 `CodeDeploy-OnPrem-App`，對於部署群組名稱使用 `CodeDeploy-OnPrem-DG`)。在您完成這些指示後，返回此頁面。

> **Note**
>
> 或者，您可以將檔案集上傳至 GitHub 存放庫，然後從該處部署它。如需詳細資訊，請參閱 CodeDeploy 與整合 GitHub。

步驟 4：部署應用程式修訂

將應用程式修訂版上傳到 Amazon S3 儲存貯體後，請嘗試將其部署到現場部署執行個體。請遵循使用建立部署 CodeDeploy中的指示，然後返回此頁面。

步驟 5：驗證您的部署

若要驗證部署是否成功，請按照 檢視 CodeDeploy 部署詳情 中的指示，然後返回此頁面。

如果部署成功，您會在`c:\temp\CodeDeployExample`資料夾中找到四個文字檔案 (適用於 Windows 伺服器) 或 `/tmp/CodeDeployExample` (適用於 Ubuntu 伺服器和 RHEL)。

如果部署失敗，請按照 View Instance Details 和 對執行個體問題進行故障診斷中的故障排除步驟進行。進行任何必要的修正，重新綁定並上傳您的應用程式修訂版，然後再試一次部署。

步驟 6：清除資源

若要避免為本教學建立的資源持續收費，請刪除 Amazon S3 儲存貯體 (如果您不再使用該儲存貯體)。您也可以清除相關聯的資源，例如中的應用程式和部署群組記錄以 CodeDeploy 及內部部署執行個體。

您可以使用 CodeDeploy 和 Amazon S3 主控台的 AWS CLI 或組合 AWS CLI 來清理資源。

清理 CLI 源

要刪除 Amazon S3 存儲桶

- 對值區呼叫 rm 指令以及`--recursive`開關 (例如，`codedeploydemobucket`)。這會刪除儲存貯體以及儲存貯體中的所有物件。

```
aws s3 rm s3://your-bucket-name --recursive --region region
```

若要刪除應用程式和部署群組記錄 CodeDeploy

- 針對應用程式呼叫刪除應用程式命令 (例如，`CodeDeploy-OnPrem-App`)。部署和部署群組的記錄將會刪除。

```
aws deploy delete-application --application-name your-application-name
```

取消註冊內部部署執行個體並刪除 IAM 使用者

- 針對內部部署執行個體和區域呼叫取消註冊命令：

```
aws deploy deregister --instance-name your-instance-name --delete-iam-user --
region your-region
```

> **Note**
>
> 如果您不想刪除與此內部部署執行個體相關聯的 IAM 使用者，請改用此`--no-delete-iam-user`選項。

解除安裝 CodeDeploy 代理程式並從內部部署執行個體移除組態檔

- 從內部部署執行個體呼叫解除安裝命令：

```
aws deploy uninstall
```

您現在已經完成所有步驟，以清除用於此教學課程的資源。

清理資源（控制台）

要刪除 Amazon S3 存儲桶

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 選擇您要刪除的儲存貯體旁的圖示 (例如 `codedeploydemobucket`)，但不選擇儲存貯體本身。
3. 選擇動作，然後選擇刪除。
4. 當提示刪除儲存貯體時，請選擇 OK (確定)。

若要刪除應用程式和部署群組記錄 CodeDeploy

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在導覽窗格中，選擇 Applications (應用程式)。
3. 選擇您要刪除的應用程式名稱 (例如，`CodeDeploy-OnPrem-App`)，然後選擇 Delete application (刪除應用程式)。
4. 當系統出現提示時，請輸入應用程式的名稱，以確認要執行刪除動作，接著選擇 Delete (刪除)。

您無法使用 AWS CodeDeploy 主控台取消註冊內部部署執行個體或解除安裝 CodeDeploy 代理程式。請遵循中的說明進行取消註冊內部部署執行個體並刪除 IAM 使用者

教學課程：用 CodeDeploy 於將應用程式部署到 Auto Scaling 群組

在本教學課程中，您將使用 CodeDeploy 將應用程式修訂部署到 Auto Scaling 群組。Amazon EC2 Auto Scaling 會使用預先定義的條件啟動 Amazon EC2 執行個體，然後在不再需要執行個體時終止這些執行個體。Amazon EC2 Auto Scaling 可確保始終具有正確數量的 Amazon EC2 執行個體可用來處理部署負載，藉此協 CodeDeploy 助擴展。如需 Amazon EC2 Auto Scaling 與整合的相關資訊 CodeDeploy，請參閱 CodeDeploy 與 Amazon EC2 Auto Scaling 集成。

主題

必要條件

在本教學課程中遵循的步驟：

- 完成中的所有步驟開始使用 CodeDeploy，包括設定和設定 AWS CLI 和建立 IAM 執行個體設定檔 (**CodeDeployDemo-EC2-Instance-Profile**) 和服務角色 (**CodeDeployDemo**)。服務角色是一種特殊類型的 IAM 角色，可提供代表您採取行動的服務權限。
- 如果您使用啟動範本建立 Auto Scaling 群組，則必須新增下列權限：
  - `ec2:RunInstances`
  - `ec2:CreateTags`
  - `iam:PassRole`

  如需詳細資訊步驟 2：建立服務角色，請參閱 Amazon EC2 Auto Scaling 使用者指南中的為自動擴展群組建立啟動範本和啟動範本支援。
- 建立並使用與 Ubuntu 伺服器執行個體和相容的修訂 CodeDeploy。對於您的修訂，您可以執行下列操作之一：
  - 在 教學課程：使用 CodeDeploy (Windows 伺服器、Ubuntu 伺服器或 RHEL) 將應用程式部署到內部部署執行個體 教學課程中建立和使用 步驟 2：建立範例應用程式修訂 中的範例修訂。

  - 若要自行建立修訂版，請參閱 使用的應用程式修訂 CodeDeploy。
- 建立以下輸入規則命名**CodeDeployDemo-AS-SG**的安全群組：
  - 類型：HTTP
  - 來源：任何地方

  這是檢視您的應用程式和驗證部署成功的必要條件。如需如何建立安全群組的詳細資訊，請參閱 Amazon EC2 使用者指南中的建立安全群組。

步驟 1：建立並設定「Auto Scaling」群組

在此步驟中，您將建立包含單一 Amazon Linux、RHEL 或 Windows 伺服器亞馬 Amazon EC2 執行個體的 Auto Scaling 群組。在稍後的步驟中，您將指示 Amazon EC2 Auto Scaling 新增一個 Amazon EC2 執行個體，並 CodeDeploy 將您的修訂部署到該執行個體。

主題

- 若要建立和設定 Auto Scaling 群組 (CLI)
- 若要建立和設定「Auto Scaling」群組 (控制台)

若要建立和設定 Auto Scaling 群組 (CLI)

1. 呼叫命create-launch-template令以建立 Amazon EC2 啟動範本。

   呼叫此命令之前，您需要適用於此教學課程之 AMI 的 ID，其由預留位置 *image-id* 代表。您還需要 Amazon EC2 執行個體 key pair 的名稱，才能存取 Amazon EC2 執行個體 (以預留位置###表示)。

   若要取得適用於此教學課程的 AMI 的 ID：

   a. 前往 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。

   b. 在導覽窗格中，在 Instances (執行個體) 下，選擇 Instances (執行個體)，然後選擇 Launch Instance (啟動執行個體)。

   c. 在「選擇 Amazon 機器映像」頁面的「快速啟動」選項卡上，請注意 Amazon Linux 2 AMI，紅帽企業 Linux 7.1，Ubuntu 服務器 14.04 LTS 或 Microsoft Windows 服務器 2012 R 2 旁邊的 AMI 的 ID。

> **Note**
>
> 如果您有與之相容的 AMI 的自訂版本，請在此處選擇它 CodeDeploy，而不是瀏覽 [快速入門] 索引標籤。如需搭配使用自訂 AMI CodeDeploy 和 Amazon EC2 自 Auto Scaling 的相關資訊，請參閱將自訂 AMI 與 CodeDeploy Amazon EC2 自 Auto Scaling 搭配使用。

對於亞馬遜 EC2 執行個體金鑰組，請使用您的 Amazon EC2 執行個體 key pair 的名稱。

呼叫 create-launch-template 命令。

在本地 Linux、macOS 電腦或 Unix 機器上：

```
aws ec2 create-launch-template \
  --launch-template-name CodeDeployDemo-AS-Launch-Template \
  --launch-template-data file://config.json
```

該config.json文件的內容：

```
{
  "InstanceType":"t1.micro",
  "ImageId":"image-id",
  "IamInstanceProfile":{
    "Name":"CodeDeployDemo-EC2-Instance-Profile"
  },
  "KeyName":"key-name"
}
```

在本機 Windows 電腦上：

```
aws ec2 create-launch-template --launch-template-name CodeDeployDemo-AS-Launch-
Template --launch-template-data file://config.json
```

該config.json文件的內容：

```
{
  "InstanceType":"t1.micro",
  "ImageId":"image-id",
```

```
  "IamInstanceProfile":{
    "Name":"CodeDeployDemo-EC2-Instance-Profile"
  },
  "KeyName":"key-name"
}
```

這些命令和config.json檔案會為您的 Auto CodeDeployDemo Scaling 群組建立名為-AS 啟動範本的 Amazon EC2 啟動範本，該範本將根據 t1.micro Amazon EC2 執行個體類型在以下步驟中建立。根據您的輸入ImageId、和 IamInstanceProfileKeyName，啟動範本還會指定 AMI ID、與啟動時傳遞給執行個體的 IAM 角色相關聯的執行個體設定檔名稱，以及連線至執行個體時要使用的 Amazon EC2 key pair。

2. 呼叫指create-auto-scaling-group令以建立「Auto Scaling」群組。您需要在 [區域] 中列出的其中一個區域中的一個可用區域名稱以及中的端點 (以預留位置###區域表示)。AWS 一般參考

> **Note**
>
> 若要檢視區域中的可用區域的清單，請呼叫：
>
> ```
> aws ec2 describe-availability-zones --region region-name
> ```
>
> 例如，若要檢視美國西部 (奧勒岡) 區域的可用區域清單，請撥打：
>
> ```
> aws ec2 describe-availability-zones --region us-west-2
> ```
>
> 有關區域名稱識別碼的清單，請參閱 依區域的資源套件時段名稱。

在本地 Linux、macOS 電腦或 Unix 機器上：

```
aws autoscaling create-auto-scaling-group \
  --auto-scaling-group-name CodeDeployDemo-AS-Group \
  --launch-template CodeDeployDemo-AS-Launch-Template,Version='$Latest' \
  --min-size 1 \
  --max-size 1 \
  --desired-capacity 1 \
  --availability-zones availability-zone \
  --tags Key=Name,Value=CodeDeployDemo,PropagateAtLaunch=true
```

在本機 Windows 電腦上：

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name
 CodeDeployDemo-AS-Group --launch-template LaunchTemplateName=CodeDeployDemo-
AS-Launch-Template,Version="$Latest" --min-size 1 --max-size 1 --
desired-capacity 1 --availability-zones availability-zone --tags
 Key=Name,Value=CodeDeployDemo,PropagateAtLaunch=true
```

這些命令會建立一個**CodeDeployDemo-AS-Group**根據名為的 Amazon EC2 啟動範本命名的自動擴展群組**CodeDeployDemo-AS-Launch-Template**。此 Auto Scaling 群組只有一個 Amazon EC2 執行個體，並在指定的可用區域中建立。此「Auto Scaling 例」群組中的每個執行個體都會有標籤Name=CodeDeployDemo。稍後安裝 CodeDeploy 代理程式時會使用此標籤。

3. 針對 **CodeDeployDemo-AS-Group** 呼叫 describe-auto-scaling-groups 命令：

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names
 CodeDeployDemo-AS-Group --query "AutoScalingGroups[0].Instances[*].[HealthStatus,
 LifecycleState]" --output text
```

在傳回值顯示 Healthy 和 InService 之前不要繼續。

4. Auto Scaling 群組中的執行個體必須安裝 CodeDeploy 代理程式，才能在 CodeDeploy 部署中使用。透過 AWS Systems Manager 使用建立 Auto Scaling 群組時新增的標籤呼叫create-association命令，以安裝 CodeDeploy 代理程式。

```
aws ssm create-association \
  --name AWS-ConfigureAWSPackage \
  --targets Key=tag:Name,Values=CodeDeployDemo \

 --parameters action=Install, name=AWSCodeDeployAgent \
  --schedule-expression "cron(0 2 ? * SUN *)"
```

此命令會在系統管理員狀態管理員中建立關聯，該關聯會將 CodeDeploy 代理程式安裝在 Auto Scaling 群組中的所有執行個體上，然後嘗試在每個星期日早上 2:00 進行更新。如需 CodeDeploy 代理程式的詳細資訊，請參閱使用 CodeDeploy 代理程式。如需有關「Systems Manager 的詳細資訊，請參閱何謂 AWS Systems Manager。

若要建立和設定「Auto Scaling」群組 (控制台)

1. 前往 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。

2. 在全域導覽列中，確定已選取「區域」中列出的其中一個區域和中AWS 一般參考的端點。Amazon EC2 Auto Scaling 資源與您指定的區域相關聯，並且 CodeDeploy 僅在特定區域提供支援。
3. 在導覽列的「執行個體」下，選擇「啟動範本」。
4. 選擇 Create launch template (建立啟動範本)。
5. 在「啟動範本名稱和描述」對話方塊中，對於 Launch 範本名稱，輸入**CodeDeployDemo-AS-Launch-Template**。保留其他欄位的預設值。
6. 在 Amazon 機器映像（AMI）對話框中，單擊 AMI 下的下拉菜單，選擇適用於本教程的 AMI：

   - 在 AMI 下拉列表的快速啟動選項卡上，選擇以下選項之一：Amazon Linux 2 AMI，紅帽企業 Linux 7.1，Ubuntu 服務器 14.04 LTS 或 Microsoft 視窗服務器 2012 R 2。

   > **Note**
   >
   > 如果您有與之相容的 AMI 的自訂版本，請在此處選擇它 CodeDeploy，而不是瀏覽 [快速入門] 索引標籤。如需搭配使用自訂 AMI CodeDeploy 和 Amazon EC2 自 Auto Scaling 的相關資訊，請參閱將自訂 AMI 與 CodeDeploy Amazon EC2 自 Auto Scaling 搭配使用。

7. 在執行個體類型中，選取下拉式清單，然後選擇 t1.micro。您可以使用搜索欄更快地找到它。
8. 在 [key pair (登入)] 對話方塊中，選取 [選擇現有金鑰配對]。在選取 key pair 下拉式清單中，選擇您在先前步驟中建立或使用的 Amazon EC2 執行個體 key pair。
9. 在 [網路設定] 對話方塊中，選擇 [虛擬公用雲端 (VPC)]。

   在 [安全性群組] 下拉式清單中，選擇您在教學課程的必要條件區段 (**CodeDeployDemo-AS-SG**) 中建立的安全性群組。
10. 展開「進階詳細資料」對話方塊。在 IAM 執行個體設定檔下拉式清單中，在 IAM 執行個體設定檔下選取您先前建立的 IAM 角色 (**CodeDeployDemo-EC2-Instance-Profile**)。

    保留其餘的預設值。
11. 選擇 Create launch template (建立啟動範本)。
12. 在「後續步驟」對話方塊中，選擇「建立 Auto Scaling」群組。
13. 在 [選擇啟動範本或組態] 頁面上，針對 [Auto Scaling] 群組名稱輸入**CodeDeployDemo-AS-Group**。
14. 在 [啟動範本] 對話方塊中，應填入您的啟動範本 (**CodeDeployDemo-AS-Launch-Template**)，如果沒有，請從下拉式功能表中選取它。保留預設值並選擇 [下一步]。

15. 在 [選擇執行個體啟動選項] 頁面的 [網路] 區段中，對於 VPC，選擇預設 VPC。然後，針對可用區域和子網路，選擇預設子網路。如果您無法選擇預設值，則必須建立 VPC。如需詳細資訊，請參閱開始使用 Amazon VPC。
16. 在 Instance type requirements (執行個體類型需求) 區段中，請使用預設設定來簡化此步驟。(請勿覆寫啟動範本。) 在本教程中，您將使用啟動範本中指定的執行個體類型，並且僅啟動隨需執行個體。
17. 選擇 Next (下一頁) 前往 Configure advanced options (設定進階選項) 頁面。
18. 保留預設值並選擇 [下一步]。
19. 在 [設定群組大小和擴展原則] 頁面上，保持預設群組大小值 1。選擇下一步。
20. 略過設定通知的步驟，然後選擇下一步。
21. 在 [新增標記] 頁面上，新增稍後安裝 CodeDeploy 代理程式時要使用的標籤。選擇 Add tag (新增標籤)。

    a. 在 Key (金鑰) 中，輸入 **Name**。

    b. 在 Value (值) 中輸入 **CodeDeployDemo**。

    選擇下一步。
22. 在「檢閱」頁面上檢閱「Auto Scaling」群組資訊，然後選擇「建立 Auto Scaling」群組。
23. 在導覽列中，選取「Auto Scaling 群組」後，選擇**CodeDeployDemo-AS-Group**，然後選擇「執行個體管理」索引標籤。直到的值InService出現在「生命週期」欄中，且「Health 全狀況」欄中顯示「狀況良好」的值之前，請勿繼續執行。
24. 依照安裝 CodeDeploy 代理程式中的步驟並使用`Name=CodeDeployDemo`執行個體標籤來安裝 CodeDeploy 代理程式。

步驟 2：將應用程式部署到「Auto Scaling」群組

在此步驟中，您將會將修訂部署到 Auto Scaling 群組中的單一 Amazon EC2 執行個體。

主題

- 建立部署 (CLI)
- 建立部署 (主控台)

建立部署 (CLI)

1. 呼叫 create-application 命令以建立名為 **SimpleDemoApp** 的應用程式：

```
aws deploy create-application --application-name SimpleDemoApp
```

2. 您應該已經遵循以下步驟 2：建立服務角色 CodeDeploy的說明建立服務角色。服務角色將 CodeDeploy 授予存取 Amazon EC2 執行個體的權限，以擴充 (讀取) 其標籤。您需要服務角色 ARN。若要取得服務角色 ARN，請遵循 取得服務角色 ARN (CLI) 中的指示。

3. 現在您已經擁有服務角色 ARN，請呼叫create-deployment-group命令，以指定的服務角色 ARN 使用名為**SimpleDemoDG**的 Auto Scaling 群組**CodeDeployDemo-AS-Group**和名為**SimpleDemoApp**的部署組態來建立名為**CodeDeployDefault.OneAtATime**的應用程式相關聯的部署群組。

> **Note**
>
> 該create-deployment-group命令支援建立觸發器，以便將 Amazon SNS 通知傳送給主題訂閱者有關部署和執行個體中指定事件的主題訂閱者。該命令還支援自動復原部署和設定警示以停止部署的選項，以便在符合 Amazon CloudWatch 警示中的監控閾值時停止部署。這些動作的指令不包括在本教學課程中。

在本地 Linux、macOS 電腦或 Unix 機器上：

```
aws deploy create-deployment-group \
  --application-name SimpleDemoApp \
  --auto-scaling-groups CodeDeployDemo-AS-Group \
  --deployment-group-name SimpleDemoDG \
  --deployment-config-name CodeDeployDefault.OneAtATime \
  --service-role-arn service-role-arn
```

在本機 Windows 電腦上：

```
aws deploy create-deployment-group --application-name SimpleDemoApp --auto-scaling-
groups CodeDeployDemo-AS-Group --deployment-group-name SimpleDemoDG --deployment-
config-name CodeDeployDefault.OneAtATime --service-role-arn service-role-arn
```

4. 使用指定位置的修訂版，呼叫 create-deployment 命令以建立與名為 **SimpleDemoApp** 的應用程式關聯的部署、名為 **CodeDeployDefault.OneAtATime** 的部署組態、名為 **SimpleDemoDG** 的部署群組。

   對於 Amazon Linux 和 RHEL Amazon EC2 執行個體，從本機 Linux、macOS 或 Unix 機器呼叫

   ```
   aws deploy create-deployment \
     --application-name SimpleDemoApp \
     --deployment-config-name CodeDeployDefault.OneAtATime \
     --deployment-group-name SimpleDemoDG \
     --s3-location bucket=bucket-name,bundleType=zip,key=samples/latest/
   SampleApp_Linux.zip
   ```

   儲存####是 Amazon S3 儲存貯體的名稱，該儲存貯體包含您所在地區的 CodeDeploy 資源套件檔案。######### （###） ############# aws-codedeploy-us-east-2如需值區名稱清單，請參閱依區域的資源套件時段名稱。

   對於 Amazon Linux 和 RHEL Amazon EC2 執行個體，從本機視窗機器呼叫

   ```
   aws deploy create-deployment --application-name SimpleDemoApp --deployment-config-
   name CodeDeployDefault.OneAtATime --deployment-group-name SimpleDemoDG --s3-
   location bucket=bucket-name,bundleType=zip,key=samples/latest/SampleApp_Linux.zip
   ```

   儲存####是 Amazon S3 儲存貯體的名稱，該儲存貯體包含您所在地區的 CodeDeploy 資源套件檔案。######### （###） ############# aws-codedeploy-us-east-2如需值區名稱清單，請參閱依區域的資源套件時段名稱。

   適用於視窗伺服器 Amazon EC2 執行個體，從本機 Linux、macOS 或 Unix 機器呼叫

   ```
   aws deploy create-deployment \
     --application-name SimpleDemoApp \
     --deployment-config-name CodeDeployDefault.OneAtATime \
     --deployment-group-name SimpleDemoDG \
     --s3-location bucket=bucket-name,bundleType=zip,key=samples/latest/
   SampleApp_Windows.zip
   ```

儲存####是 Amazon S3 儲存貯體的名稱，該儲存貯體包含您所在地區的 CodeDeploy 資源套件檔案。######### （###） ############ aws-codedeploy-us-east-2如需值區名稱清單，請參閱依區域的資源套件時段名稱。

對於視窗伺服器 Amazon EC2 執行個體，從本機視窗電腦呼叫

```
aws deploy create-deployment --application-name SimpleDemoApp --deployment-config-
name CodeDeployDefault.OneAtATime --deployment-group-name SimpleDemoDG --s3-
location bucket=bucket-name,bundleType=zip,key=samples/latest/SampleApp_Windows.zip
```

儲存####是 Amazon S3 儲存貯體的名稱，該儲存貯體包含您所在地區的 CodeDeploy 資源套件檔案。######### （###） ############ aws-codedeploy-us-east-2如需值區名稱清單，請參閱依區域的資源套件時段名稱。

> **Note**
>
> 目前，尚 CodeDeploy 未提供範例修訂版來部署到 Ubuntu 伺服器的 Amazon EC2 執行個體。若要自行建立修訂版，請參閱使用的應用程式修訂 CodeDeploy

5. 呼叫 get-deployment 命令，確保部署成功。

呼叫此命令之前，您需要部署的 ID，其應該已由呼叫傳回 create-deployment 命令。 如果您需要再次取得部署 ID，請針對名為 **SimpleDemoApp** 的應用程式與名為 **SimpleDemoDG** 的部署群組呼叫 list-deployments 命令。

```
aws deploy list-deployments --application-name SimpleDemoApp --deployment-group-
name SimpleDemoDG --query "deployments" --output text
```

現在，利用部署 ID 呼叫 get-deployment 命令。

```
aws deploy get-deployment --deployment-id deployment-id --query
 "deploymentInfo.status" --output text
```

在傳回的值為 Succeeded 之前不要繼續。

建立部署 (主控台)

1. 您應該已經遵循以下步驟 2：建立服務角色 CodeDeploy的說明建立服務角色。服務角色將 CodeDeploy 授予存取執行個體以擴充 (讀取) 其標記的權限。使用 CodeDeploy 主控台部署應用程式修訂版本之前，您將需要服務角色 ARN。若要取得服務角色 ARN，請遵循 取得服務角色 ARN (主控台) 中的指示。
2. 現在您已經擁有服務角色 ARN，您可以使用 CodeDeploy 主控台來部署應用程式修訂版本。

   請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

3. 在瀏覽窗格中，展開 [部署]，然後選擇 [應用程式]。
4. 選擇建立應用程式。
5. 選擇 Custom application (自訂應用程式)。
6. 在 Application name (應用程式名稱) 中，輸入 **SimpleDemoApp**。
7. 在 Compute Platform (運算平台) 中，選擇 EC2/On-premises (EC2/現場部署)。
8. 選擇建立應用程式。
9. 在 Deployment groups (部署群組) 標籤中，選擇 Create deployment group (建立部署群組)。
10. 在 Deployment group name (部署群組名稱) 中，輸入 **SimpleDemoDG**。
11. 在 Service Role (服務角色) 中，選擇您服務角色的名稱。
12. 在 Deployment type (部署類型) 中，選擇 In-place (就地)。
13. 在 [環境設定] 中，選取 [Auto Scaling] 群組，然後選擇**CodeDeployDemo-AS-Group**。
14. 在 [部署組態] 中，選擇CodeDeployDefault。 OneAtA 時間。
15. 清除 Enable load balancing (啟用負載平衡)。
16. 選擇 Create deployment group (建立部署群組)。
17. 在部署群組標籤中，選擇 Create deployment (建立部署)。
18. 在 [修訂版本類型] 中，選擇 [我的應用程式存放在 Amazon S3]。
19. 在 Revision location (修訂版位置)，輸入作業系統和區域的範例應用程式的位置。

    適用於 Amazon Linux 和 RHEL Amazon EC2 執行個體

區域	範例應用程式的位置
美國東部 (俄亥俄) 區域	http://s3-us-east-2.amazonaws.com/aws-codedeploy-us-east-2/samples/latest/SampleApp_Linux.zip
美國東部 (維吉尼亞北部) 區域	http://s3.amazonaws.com/aws-codedeploy-us-east-1/samples/latest/SampleApp_Linux.zip
美國西部 (加利佛尼亞北部) 區域	http://s3-us-west-1.amazonaws.com/aws-codedeploy-us-west-1/samples/latest/SampleApp_Linux.zip
美國西部 (奧勒岡) 區域	http://s3-us-west-2.amazonaws.com/aws-codedeploy-us-west-2/samples/latest/SampleApp_Linux.zip
加拿大 (中部) 區域	http://s3-ca-central-1.amazonaws.com/aws-codedeploy-ca-central-1/samples/latest/SampleApp_Linux.zip
歐洲 (愛爾蘭) 區域	http://s3-eu-west-1.amazonaws.com/aws-codedeploy-eu-west-1/samples/latest/SampleApp_Linux.zip
歐洲 (倫敦) 區域	http://s3-eu-west-2.amazonaws.com/aws-codedeploy-eu-west-2/samples/latest/SampleApp_Linux.zip

區域	範例應用程式的位置
歐洲 (巴黎) 區域	http://s3-eu-west-3.amazonaws.com/aws-codedeploy-eu-west-3/samples/latest/SampleApp_Linux.zip
歐洲 (法蘭克福) 區域	http://s3-eu-central-1.amazonaws.com/aws-codedeploy-eu-central-1/samples/latest/SampleApp_Linux.zip
以色列 (特拉維夫) 區域	https://aws-codedeploy-il-central-1.s3.il-central-1.amazonaws.com/samples/latest/SampleApp_Linux.zip
亞太區域 (香港) 區域	https://aws-codedeploy-ap-east-1.s3.ap-east-1.amazonaws.com/samples/latest/SampleApp_Linux.zip
亞太區域 (東京) 區域	http://s3-ap-northeast-1.amazonaws.com/aws-codedeploy-ap-northeast-1/samples/latest/SampleApp_Linux.zip
亞太區域 (首爾) 區域	http://s3-ap-northeast-2.amazonaws.com/aws-codedeploy-ap-northeast-2/samples/latest/SampleApp_Linux.zip
亞太區域 (新加坡) 區域	http://s3-ap-southeast-1.amazonaws.com/aws-codedeploy-ap-southeast-1/samples/latest/SampleApp_Linux.zip

區域	範例應用程式的位置
亞太區域 (雪梨) 區域	http://s3-ap-southeast-2.amazonaws.com/aws-codedeploy-ap-southeast-2/samples/latest/SampleApp_Linux.zip
亞太區域 (墨爾本) 區域	https://aws-codedeploy-ap-southeast-4.s3.ap-southeast-4.amazonaws.com/samples/latest/SampleApp_Linux.zip
亞太 (孟買) 區域	http://s3-ap-south-1.amazonaws.com/aws-codedeploy-ap-south-1/samples/latest/SampleApp_Linux.zip
南美洲 (聖保羅) 區域	http://s3-sa-east-1.amazonaws.com/aws-codedeploy-sa-east-1/samples/latest/SampleApp_Linux.zip

對於視窗服務器 Amazon EC2 實例

區域	範例應用程式的位置
美國東部 (俄亥俄) 區域	http://s3-us-east-2.amazonaws.com/aws-codedeploy-us-east-2/samples/latest/SampleApp_Windows.zip
美國東部 (維吉尼亞北部) 區域	http://s3.amazonaws.com/aws-codedeploy-us-east-1/samples/latest/SampleApp_Windows.zip

區域	範例應用程式的位置
美國西部 (加利佛尼亞北部) 區域	http://s3-us-west-1.amazonaws.com/aws-codedeploy-us-west-1/samples/latest/SampleApp_Windows.zip
美國西部 (奧勒岡) 區域	http://s3-us-west-2.amazonaws.com/aws-codedeploy-us-west-2/samples/latest/SampleApp_Windows.zip
加拿大 (中部) 區域	http://s3-ca-central-1.amazonaws.com/aws-codedeploy-ca-central-1/samples/latest/SampleApp_Windows.zip
歐洲 (愛爾蘭) 區域	http://s3-eu-west-1.amazonaws.com/aws-codedeploy-eu-west-1/samples/latest/SampleApp_Windows.zip
歐洲 (倫敦) 區域	http://s3-eu-west-2.amazonaws.com/aws-codedeploy-eu-west-2/samples/latest/SampleApp_Windows.zip
歐洲 (巴黎) 區域	http://s3-eu-west-3.amazonaws.com/aws-codedeploy-eu-west-3/samples/latest/SampleApp_Windows.zip
歐洲 (法蘭克福) 區域	http://s3-eu-central-1.amazonaws.com/aws-codedeploy-eu-central-1/samples/latest/SampleApp_Windows.zip

區域	範例應用程式的位置
以色列 (特拉維夫) 區域	https://aws-codedeploy-il-central-1.s3.il-central-1.amazonaws.com/samples/latest/SampleApp_Windows.zip
亞太區域 (香港) 區域	https://aws-codedeploy-ap-east-1.s3.ap-east-1.amazonaws.com/samples/latest/SampleApp_Windows.zip
亞太 (首爾) 區域	http://s3-ap-northeast-2.amazonaws.com/aws-codedeploy-ap-northeast-2/samples/latest/SampleApp_Windows.zip
亞太區域 (新加坡) 區域	http://s3-ap-southeast-1.amazonaws.com/aws-codedeploy-ap-southeast-1/samples/latest/SampleApp_Windows.zip
亞太區域 (雪梨) 區域	http://s3-ap-southeast-2.amazonaws.com/aws-codedeploy-ap-southeast-2/samples/latest/SampleApp_Windows.zip
亞太區域 (墨爾本) 區域	https://aws-codedeploy-ap-southeast-4.s3.ap-southeast-4.amazonaws.com/samples/latest/SampleApp_Windows.zip
亞太 (孟買) 區域	http://s3-ap-south-1.amazonaws.com/aws-codedeploy-ap-south-1/samples/latest/SampleApp_Windows.zip

區域	範例應用程式的位置
南美洲 (聖保羅) 區域	`http://s3-sa-east-1.amazonaws.com/aws-codedeploy-sa-east-1/samples/latest/SampleApp_Windows.zip`

對於 Ubuntu 服務器 Amazon EC2 實例

輸入存放在 Amazon S3 中的自訂應用程式修訂版本的位置。

20. 將 Deployment description (部署描述) 空白。
21. 展開 Advanced (進階)。
22. 選擇 Create deployment (建立部署)。

> **Note**
>
> 如果狀態出現 Failed (失敗)，而非 Succeeded (成功)，則可嘗試監控和疑難排解您的部署中的某些技術 (使用 **SimpleDemoApp** 應用程式名稱，以及 **SimpleDemoDG** 部署群組名稱)。

步驟 3：檢查結果

在此步驟中，您將檢查是否已在 Auto Scaling 群組中的單一 Amazon EC2 執行個體上 CodeDeploy 安裝了**SimpleDemoApp**修訂版。

主題

- 檢查結果 (CLI)
- 檢查結果 (主控台)

檢查結果 (CLI)

首先，您需要 Amazon EC2 實例的公共 DNS。

使用呼叫describe-instances命令 AWS CLI 來取得 Auto Scaling 群組中 Amazon EC2 執行個體的公有 DNS。

在呼叫此命令之前，您需要 Amazon EC2 執行個體的 ID。若要取得 ID，可如您之前的做法針對 **CodeDeployDemo-AS-Group** 呼叫 describe-auto-scaling-groups 命令：

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names CodeDeployDemo-AS-Group --query "AutoScalingGroups[0].Instances[*].InstanceId" --output text
```

現在呼叫 describe-instances 命令。

```
aws ec2 describe-instances --instance-id instance-id --query
 "Reservations[0].Instances[0].PublicDnsName" --output text
```

傳回的值是亞馬遜 EC2 執行個體的公有 DNS。

使用網頁瀏覽器，使用如下所示的 URL 顯示部署到該 Amazon EC2 執行個體的 SimpleDemoApp 修訂版本：

```
http://ec2-01-234-567-890.compute-1.amazonaws.com
```

如果您看到祝賀頁面，表示您已成功將修訂版部署 CodeDeploy 到 Auto Scaling 群組中的單一 Amazon EC2 執行個體！

接下來，您將新增一個 Amazon EC2 執行個體到 Auto Scaling 群組。在 Amazon EC2 Auto Scaling 新增 Amazon EC2 執行個體之後， CodeDeploy 會將您的修訂版部署到新的執行個體。

檢查結果 (主控台)

首先，您需要 Amazon EC2 實例的公共 DNS。

前往 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。

在 Amazon EC2 導覽窗格的「Auto Scaling」下，選擇「Auto Scaling 群組」，然後選擇**CodeDeployDemo-AS-Group**項目。

在執行個體索引標籤上，選擇清單中的 Amazon EC2 執行個體 ID。

在 Instances (執行個體) 頁面上，於Description (描述) 標籤上，記下 Public DNS (公開 DNS) 值。 其看起來如下所示：**ec2-01-234-567-890.compute-1.amazonaws.com**

使用網頁瀏覽器，使用如下所示的 URL 顯示部署到該 Amazon EC2 執行個體的 SimpleDemoApp 修訂版本：

```
http://ec2-01-234-567-890.compute-1.amazonaws.com
```

如果您看到祝賀頁面，表示您已成功將修訂版部署 CodeDeploy 到 Auto Scaling 群組中的單一 Amazon EC2 執行個體！

接下來，您將 Amazon EC2 執行個體新增至 Auto Scaling 群組。在 Amazon EC2 Auto Scaling 新增 Amazon EC2 執行個體之後， CodeDeploy 會將您的修訂版部署到新的 Amazon EC2 執行個體。

步驟 4：增加 Auto Scaling 群組中 Amazon EC2 執行個體的數量

在此步驟中，您要指示 Auto Scaling 群組建立額外的 Amazon EC2 執行個體。在 Amazon EC2 Auto Scaling 建立執行個體之後， CodeDeploy 將您的修訂部署到該執行個體。

主題

- Auto Scaling 群組 (CLI) 中 Amazon EC2 執行個體的數量
- 擴展部署群組 (主控台) 中 Amazon EC2 執行個體的數量

Auto Scaling 群組 (CLI) 中 Amazon EC2 執行個體的數量

1. 呼叫命update-auto-scaling-group令，將 Auto Scaling 群組中名為的 Amazon EC2 執行個體**CodeDeployDemo-AS-Group**從一個增加到兩個。

   在本地 Linux、macOS 電腦或 Unix 機器上：

   ```
   aws autoscaling update-auto-scaling-group \
     --auto-scaling-group-name CodeDeployDemo-AS-Group \
     --min-size 2 \
     --max-size 2 \
     --desired-capacity 2
   ```

   在本機 Windows 電腦上：

   ```
   aws autoscaling update-auto-scaling-group --auto-scaling-group-name CodeDeployDemo-
   AS-Group --min-size 2 --max-size 2 --desired-capacity 2
   ```

2. 確定 Auto Scaling 群組現在有兩個 Amazon EC2 執行個體。針對 **CodeDeployDemo-AS-Group** 呼叫 describe-auto-scaling-groups 命令：

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names
 CodeDeployDemo-AS-Group --query "AutoScalingGroups[0].Instances[*].[HealthStatus,
 LifecycleState]" --output text
```

在兩個傳回值顯示 Healthy 和 InService 之前不要繼續。

擴展部署群組 (主控台) 中 Amazon EC2 執行個體的數量

1. 在 Amazon EC2 導覽列的「Auto Scaling」下，選擇「Auto Scaling 群組」，然後選擇**CodeDeployDemo-AS-Group**。
2. 選擇動作，然後選擇編輯。
3. 在 Details (詳細資訊) 標籤，在 Desired (所需)、Min (最小) 和 Max (最大) 方塊中，輸入 **2**，然後選擇 Save (儲存)。
4. 選擇執行個體標籤。新的 Amazon EC2 執行個體應該會出現在清單中。(如果執行個體沒有顯示，您也許需要選擇 Refresh (重新整理) 按鈕數次)。直到的值InService出現在「生命週期」欄中，且「Health 全狀況」欄中顯示「狀況良好」的值之前，請勿繼續執行。

步驟 5：再次檢查結果

在這個步驟中，您將檢查 Auto Scaling 群組中的新執行個體是否 CodeDeploy 已安裝 SimpleDemoApp 修訂版本。

主題

- 檢查自動部署結果 (CLI)
- 檢查自動部署結果 (主控台)

檢查自動部署結果 (CLI)

1. 在呼叫 get-deployment 命令之前，您將需要自動部署的 ID。取得 ID 後，針對名為 **SimpleDemoApp** 的應用程式及名為 **SimpleDemoDG** 的部署群組呼叫 list-deployments 命令。

```
aws deploy list-deployments --application-name SimpleDemoApp --deployment-group-
name SimpleDemoDG --query "deployments" --output text
```

應該會有兩個部署 ID。使用您還沒有用於呼叫 get-deployment 的命令：

```
aws deploy get-deployment --deployment-id deployment-id --query "deploymentInfo.
[status, creator]" --output text
```

除了部署狀態之外，您應該autoScaling在命令輸出中看到。（autoScaling表示 Amazon EC2 Auto Scaling 創建了部署。）

直到部署狀態顯示 Succeeded 之前，請勿繼續。

2. 在呼叫describe-instances命令之前，您需要新 Amazon EC2 執行個體的識別碼。若要取得此 ID，請再次針對 **CodeDeployDemo-AS-Group** 呼叫 describe-auto-scaling-groups 命令。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names
 CodeDeployDemo-AS-Group --query "AutoScalingGroups[0].Instances[*].InstanceId" --
output text
```

現在呼叫 describe-instances 命令：

```
aws ec2 describe-instances --instance-id instance-id --query
 "Reservations[0].Instances[0].PublicDnsName" --output text
```

在describe-instances命令的輸出中，記下新 Amazon EC2 執行個體的公有 DNS。

3. 使用網頁瀏覽器，使用如下所示的 URL 顯示部署到該 Amazon EC2 執行個體的SimpleDemoApp修訂版本：

```
http://ec2-01-234-567-890.compute-1.amazonaws.com
```

如果出現祝賀頁面，表示您已經在 Au CodeDeploy to Scaling 群組中向上擴展的 Amazon EC2 執行個體部署修訂版本！

檢查自動部署結果 (主控台)

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

> **Note**
>
> 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在瀏覽窗格中，展開 [部署]，然後選擇 [部署]。

3. 選擇 Amazon EC2 Auto Scaling 所建立之部署的部署識別碼。

   .

4. 此 Deployment (部署) 頁面會顯示有關部署的資訊。一般而言，您可以自行建立部署，但 Amazon EC2 Auto Scaling 會代表您建立一個部署，以便將修訂版部署到新的 Amazon EC2 執行個體。
5. 在頁面頂端顯示 Succeeded (成功) 後，在執行個體上驗證結果。首先，您需要取得執行個體的公有 DNS：
6. 在 Amazon EC2 導覽窗格的「Auto Scaling」下，選擇「Auto Scaling 群組」，然後選擇**CodeDeployDemo-AS-Group**項目。
7. 在執行個體索引標籤上，選擇新 Amazon EC2 執行個體的識別碼。
8. 在 Instances (執行個體) 頁面上，於Description (描述) 標籤上，記下 Public DNS (公開 DNS) 值。其看起來如下所示：**ec2-01-234-567-890.compute-1.amazonaws.com**

使用如下的 URL，顯示部署到執行個體的 `SimpleDemoApp` 修訂版：

```
http://ec2-01-234-567-890.compute-1.amazonaws.com
```

如果出現祝賀頁面，表示您已經在 Au CodeDeploy to Scaling 群組中向上擴展的 Amazon EC2 執行個體部署修訂版本！

步驟 6：清除

在此步驟中，您將刪除 Auto Scaling 群組，以避免您在本教學課程中使用的資源持續收費。您可以選擇性地刪除 Auto Scaling 組態和 CodeDeploy部署元件記錄。

主題

- 清除資源 (CLI)
- 清除資源 (主控台)

清除資源 (CLI)

1. 透過呼叫delete-auto-scaling-group指令來刪除「Auto Scaling」群組**CodeDeployDemo-AS-Group**。這也會終止 Amazon EC2 執行個體。

```
aws autoscaling delete-auto-scaling-group --auto-scaling-group-name CodeDeployDemo-AS-Group --force-delete
```

2. 或者，您也可以針對名為下列的啟動組態呼叫delete-launch-template指令，以刪除 Auto Scaling 啟動範本**CodeDeployDemo-AS-Launch-Template**：

```
aws ec2 delete-launch-template --launch-template-name CodeDeployDemo-AS-Launch-Template
```

3. 您可以選擇性地針對名為的 CodeDeploy 應用程式呼叫delete-application命令來刪除應用程式**SimpleDemoApp**。這將刪除所有相關聯的部署、部署群組及修訂記錄。

```
aws deploy delete-application --application-name SimpleDemoApp
```

4. 若要刪除「Systems Manager 狀態管理員」關聯，請呼叫delete-association指令。

```
aws ssm delete-association --assocation-id association-id
```

您可以通過調用命###### *ID*。describe-association

```
aws ssm describe-association --name AWS-ConfigureAWSPackage --targets
 Key=tag:Name,Values=CodeDeployDemo
```

清除資源 (主控台)

若要刪除同時終止 Amazon EC2 執行個體的 Auto Scaling 群組：

1. 登入 AWS Management Console 並開啟 Amazon EC2 主控台，網址為 https://console.aws.amazon.com/ec2/。
2. 在 Amazon EC2 導覽窗格的「Auto Scaling」下，選擇「Auto Scaling 群組」，然後選擇**CodeDeployDemo-AS-Group**項目。
3. 依序選擇 Actions (動作)、Delete (刪除) 和 Yes, Delete (是，刪除)。

(選擇性) 若要刪除啟動範本：

1. 在導覽列的「Auto Scaling」下，選擇「啟動組態」，然後選擇**CodeDeployDemo-AS-Launch-Template**。
2. 依序選擇 Actions (執行)、Delete launch configuration (刪除啟動組態) 和 Yes, Delete (是，刪除)。

1. 您也可以從中刪除應用程式 CodeDeploy。這將刪除所有相關聯的部署、部署群組及修訂記錄。請在以下位置開啟 CodeDeploy 主控台。 https://console.aws.amazon.com/codedeploy
2. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

   在瀏覽窗格中，展開 [部署]，然後選擇 [應用程式]。

3. 在應用程式清單中，選擇SimpleDemoApp。
4. 在 Application details (應用程式詳細資訊) 頁面上，選擇 Delete application (刪除應用程式)。
5. 當出現提示時，輸入 **Delete**，然後選擇 Delete (刪除)。

若要刪除「系 Systems Manager 狀態管理員」關聯：

1. 請在以下位置開啟 AWS Systems Manager 主控台。https://console.aws.amazon.com/systems-manager
2. 在導覽窗格中，選擇 State Manager (狀態管理員)。
3. 選擇您建立的關聯，然後選擇 Delete (刪除)。

教學課程：用 CodeDeploy 來部署應用程式 GitHub

在本教學中，您可 CodeDeploy 以使用將應用程式修訂範例部署 GitHub 到執行 Amazon Linux 的單一 Amazon EC2 執行個體、單一 RHEL (RHEL) 執行個體或單一 Windows 伺服器執行個體。如需與 GitHub 整合的相關資訊 CodeDeploy，請參閱 CodeDeploy 與整合 GitHub。

> **Note**
>
> 您也可以使用 CodeDeploy 將應用程式修訂從部署 GitHub 到 Ubuntu 伺服器執行個體。您可以使用中步驟 2：建立範例應用程式修訂描述的版本修訂範例教學課程：使用 CodeDeploy (Windows 伺服器、Ubuntu 伺服器或 RHEL) 將應用程式部署到內部部署執行個體，也可以建立與 Ubuntu Server 執行個體和相容的版本修訂 CodeDeploy。若要建立您自己的修訂版，請參閱規劃修訂 CodeDeploy和將應用程式規格檔案新增至修訂 CodeDeploy

主題

必要條件

在開始本教學課程之前，執行以下作業：

- 在您的本機電腦上安裝 Git。若要安裝 Git，請參閱 Git 下載。
- 完成 開始使用 CodeDeploy 中的步驟，包含安裝及設定 AWS CLI。如果您想要使用將修訂部署 AWS CLI 至執行個體，這一點特別重 GitHub 要。

步驟 1：設定 GitHub帳戶

您將需要一個 GitHub 帳戶來創建一個存儲 GitHub 庫，該存儲庫修訂將被存儲。如果您已經有 GitHub 帳戶，請跳至步驟 2：建立 GitHub儲存庫。

1. 移至 https://github.com/join。
2. 輸入使用者名稱、您的電子郵件地址，以及一個密碼。

3. 選擇 [註冊] GitHub，然後依照指示進行。

步驟 2：建立 GitHub儲存庫

您將需要一個 GitHub 存儲庫來存儲修訂。

如果您已經有 GitHub 儲存庫，請務必**CodeDeployGitHubDemo**在本教學課程中取代其名稱，然後跳到步驟 3：將範例應用程式上傳至您的 GitHub 儲存庫。

1. 在GitHub 首頁上，執行下列其中一項作業：
   - 在 Your repositories (您的儲存庫)，選擇 New repository (新的儲存庫)。
   - 在瀏覽列上，選擇 Create new (新建) (+)，然後選擇 New repository (新的存放庫)。
2. 在 Create a new repository (建立新的存放庫) 網頁中，執行下列動作：
   - 在 Repository name (儲存庫名稱) 方塊中，輸入 **CodeDeployGitHubDemo**。
   - 選取 Public (公有)。

     > **Note**
     >
     > 選取預設的 Public (公有) 選項，表示任何人都可以查看這個儲存庫。您可以選取 Private (私有) 選項，以限制誰可以查看和遞交到儲存庫。

   - 清除 Initialize this repository with a README (以 README 初始化這個儲存庫) 核取方塊。您將在下一個階段手動建立一個 `README.md` 檔案。
   - 選擇建立儲存庫。
3. 依照您的本機指示輸入命令列，以建立儲存庫。

   > **Note**
   >
   > 如果您已啟用雙重認證 GitHub，請確保在系統提示輸入密碼時輸入您的個人訪問令牌而不是 GitHub 登錄密碼。如需詳細資訊，請參閱提供您的 2FA 驗證碼。

在本地 Linux、macOS 電腦或 Unix 機器上：

1. 在終端機上，一次執行一個以下命令，其中 user *name* 是您的 GitHub 使用者名稱：

```
mkdir /tmp/CodeDeployGitHubDemo
```

```
cd /tmp/CodeDeployGitHubDemo
```

```
touch README.md
```

```
git init
```

```
git add README.md
```

```
git commit -m "My first commit"
```

```
git remote add origin https://github.com/user-name/CodeDeployGitHubDemo.git
```

```
git push -u origin master
```

2. 在 /tmp/CodeDeployGitHubDemo 位置使終端機保持開啟。

在本機 Windows 電腦上：

1. 從命令提示字元以管理員身分執行執行下列命令，一次一個：

```
mkdir c:\temp\CodeDeployGitHubDemo
```

```
cd c:\temp\CodeDeployGitHubDemo
```

```
notepad README.md
```

2. 在記事本中，儲存 README.md 檔案。關閉記事本。每次執行一個指令，其中 user *name* *####* GitHub 用者名稱：

```
git init
```

```
git add README.md
```

```
git commit -m "My first commit"
```

```
git remote add origin https://github.com/user-name/CodeDeployGitHubDemo.git
```

```
git push -u origin master
```

3. 在 c:\temp\CodeDeployGitHubDemo位置中使命令提示字元保持開啟。

步驟 3：將範例應用程式上傳至您的 GitHub 儲存庫

在此步驟中，您會將範例修訂從公有 Amazon S3 儲存貯體複製到儲 GitHub存庫。(為了簡化，對於此教學課程提供的範例修訂版是單一網頁)。

> **Note**
>
> 如果您使用其中一個修訂版，而不是我們的範例修訂版，您的修訂版必須：
>
> - 遵循 規劃修訂 CodeDeploy 和 將應用程式規格檔案新增至修訂 CodeDeploy 中的方針。
> - 使用對應的執行個體類型。
> - 可以從您的 GitHub 儀表板訪問。
>
> 如果您的修訂版符合這些要求，請直接跳到步驟 5：建立應用程式和部署群組。
> 如果您要部署到 Ubuntu 伺服器執行個體，則需要將 GitHub與 Ubuntu 伺服器執行個體和 CodeDeploy. 如需詳細資訊，請參閱 規劃修訂 CodeDeploy 及 將應用程式規格檔案新增至修訂 CodeDeploy。

主題

- 從本機 Linux、macOS 或 Unix 機器推送範例修訂版本
- 從本機 Windows 電腦推送範例修訂版

從本機 Linux、macOS 或 Unix 機器推送範例修訂版本

您的終端機仍然開啟，例如 /tmp/CodeDeployGitHubDemo 位置，請一次執行以下一個命令：

> **Note**
>
> 如果您計劃部署到 Windows 伺服器執行個體，SampleApp_Windows.zip請SampleApp_Linux.zip在命令中取代。

```
(Amazon S3 copy command)
```

```
unzip SampleApp_Linux.zip
```

```
rm SampleApp_Linux.zip
```

```
git add .
```

```
git commit -m "Added sample app"
```

```
git push
```

其中*#Amazon S3 #####*是以下之一：

- aws s3 cp s3://aws-codedeploy-us-east-2/samples/latest/SampleApp_Linux.zip . --region us-east-2美國東部 (俄亥俄) 地區
- aws s3 cp s3://aws-codedeploy-us-east-1/samples/latest/SampleApp_Linux.zip . --region us-east-1美國東部 (維吉尼亞北部) 區域
- aws s3 cp s3://aws-codedeploy-us-west-1/samples/latest/SampleApp_Linux.zip . --region us-west-1美國西部 (加利佛尼亞北部) 區域
- aws s3 cp s3://aws-codedeploy-us-west-2/samples/latest/SampleApp_Linux.zip . --region us-west-2美國西部 (奧勒岡) 區域
- aws s3 cp s3://aws-codedeploy-ca-central-1/samples/latest/SampleApp_Linux.zip . --region ca-central-1加拿大 (中部) 地區

- aws s3 cp s3://aws-codedeploy-eu-west-1/samples/latest/SampleApp_Linux.zip . --region eu-west-1歐洲 (愛爾蘭) 地區
- aws s3 cp s3://aws-codedeploy-eu-west-2/samples/latest/SampleApp_Linux.zip . --region eu-west-2歐洲 (倫敦) 地區
- aws s3 cp s3://aws-codedeploy-eu-west-3/samples/latest/SampleApp_Linux.zip . --region eu-west-3歐洲 (巴黎) 地區
- aws s3 cp s3://aws-codedeploy-eu-central-1/samples/latest/SampleApp_Linux.zip . --region eu-central-1歐洲 (法蘭克福) 地區
- aws s3 cp s3://aws-codedeploy-il-central-1/samples/latest/SampleApp_Linux.zip . --region il-central-1以色列 (特拉維夫) 地區
- aws s3 cp s3://aws-codedeploy-ap-east-1/samples/latest/SampleApp_Linux.zip . --region ap-east-1亞太區域 (香港) 地區
- aws s3 cp s3://aws-codedeploy-ap-northeast-1/samples/latest/SampleApp_Linux.zip . --region ap-northeast-1亞太區域 (東京) 地區
- aws s3 cp s3://aws-codedeploy-ap-northeast-2/samples/latest/SampleApp_Linux.zip . --region ap-northeast-2亞太區域 (首爾) 地區
- aws s3 cp s3://aws-codedeploy-ap-southeast-1/samples/latest/SampleApp_Linux.zip . --region ap-southeast-1亞太區域 (新加坡) 區域
- aws s3 cp s3://aws-codedeploy-ap-southeast-2/samples/latest/SampleApp_Linux.zip . --region ap-southeast-2亞太區域 (雪梨) 地區
- aws s3 cp s3://aws-codedeploy-ap-southeast-4/samples/latest/SampleApp_Linux.zip . --region ap-southeast-4亞太區域 (墨爾本) 地區
- aws s3 cp s3://aws-codedeploy-ap-south-1/samples/latest/SampleApp_Linux.zip . --region ap-south-1亞太區域 (孟買) 地區
- aws s3 cp s3://aws-codedeploy-sa-east-1/samples/latest/SampleApp_Linux.zip . --region sa-east-1南美洲 (聖保羅) 地區

從本機 Windows 電腦推送範例修訂版

您的命令提示字元仍然開啟，例如 c:\temp\CodeDeployGitHubDemo 位置，請一次執行以下一個命令：

> **Note**
>
> 如果您計劃部署到 Amazon Linux 或 RHEL 執行個體，請在命令SampleApp_Windows.zip中取代SampleApp_Linux.zip。

```
(Amazon S3 copy command)
```

直接解壓縮 ZIP 檔案的 the 內容到本機目錄 (例如 c:\temp\CodeDeployGitHubDemo)，而不是到新的子目錄。

```
git add .
```

```
git commit -m "Added sample app"
```

```
git push
```

其中*#Amazon S3 #####*是以下之一：

- aws s3 cp s3://aws-codedeploy-us-east-2/samples/latest/SampleApp_Windows.zip . --region us-east-2美國東部 (俄亥俄) 地區
- aws s3 cp s3://aws-codedeploy-us-east-1/samples/latest/SampleApp_Windows.zip . --region us-east-1美國東部 (維吉尼亞北部) 區域
- aws s3 cp s3://aws-codedeploy-us-west-1/samples/latest/SampleApp_Windows.zip . --region us-west-1美國西部 (加利佛尼亞北部) 區域
- aws s3 cp s3://aws-codedeploy-us-west-2/samples/latest/SampleApp_Windows.zip . --region us-west-2美國西部 (奧勒岡) 區域
- aws s3 cp s3://aws-codedeploy-ca-central-1/samples/latest/SampleApp_Windows.zip . --region ca-central-1加拿大 (中部) 地區
- aws s3 cp s3://aws-codedeploy-eu-west-1/samples/latest/SampleApp_Windows.zip . --region eu-west-1歐洲 (愛爾蘭) 地區
- aws s3 cp s3://aws-codedeploy-eu-west-2/samples/latest/SampleApp_Windows.zip . --region eu-west-2歐洲 (倫敦) 地區
- aws s3 cp s3://aws-codedeploy-eu-west-3/samples/latest/SampleApp_Windows.zip . --region eu-west-3歐洲 (巴黎) 地區

- `aws s3 cp s3://aws-codedeploy-eu-central-1/samples/latest/SampleApp_Windows.zip . --region eu-central-1`歐洲 (法蘭克福) 地區
- `aws s3 cp s3://aws-codedeploy-il-central-1/samples/latest/SampleApp_Windows.zip . --region il-central-1`以色列 (特拉維夫) 地區
- `aws s3 cp s3://aws-codedeploy-ap-east-1/samples/latest/SampleApp_Windows.zip . --region ap-east-1`亞太區域 (香港) 地區
- `aws s3 cp s3://aws-codedeploy-ap-northeast-1/samples/latest/SampleApp_Windows.zip . --region ap-northeast-1`亞太區域 (東京) 地區
- `aws s3 cp s3://aws-codedeploy-ap-northeast-2/samples/latest/SampleApp_Windows.zip . --region ap-northeast-2`亞太區域 (首爾) 地區
- `aws s3 cp s3://aws-codedeploy-ap-southeast-1/samples/latest/SampleApp_Windows.zip . --region ap-southeast-1`亞太區域 (新加坡) 區域
- `aws s3 cp s3://aws-codedeploy-ap-southeast-2/samples/latest/SampleApp_Windows.zip . --region ap-southeast-2`亞太區域 (雪梨) 地區
- `aws s3 cp s3://aws-codedeploy-ap-southeast-4/samples/latest/SampleApp_Windows.zip . --region ap-southeast-4`亞太區域 (墨爾本) 地區
- `aws s3 cp s3://aws-codedeploy-ap-south-1/samples/latest/SampleApp_Windows.zip . --region ap-south-1`亞太區域 (孟買) 地區
- `aws s3 cp s3://aws-codedeploy-sa-east-1/samples/latest/SampleApp_Windows.zip . --region sa-east-1`南美洲 (聖保羅) 地區

若要將您自己的修訂版推送至 Ubuntu Server 執行個體，請將您的修訂複製到本機存放庫中，然後呼叫下列命令：

```
git add .
git commit -m "Added Ubuntu app"
git push
```

步驟 4：佈建執行個體

在此步驟中，您會建立或設定將部署範本應用程式的執行個體。您可以部署到 Amazon EC2 執行個體或執行其中一個支援的作業系統的現場部署執行個體 CodeDeploy。如需相關資訊，請參閱 CodeDeploy 代理程式支援的作業系統。(如果您已設定要用於 CodeDeploy 部署的執行個體，請跳至下一個步驟。)

若要佈建執行個體

1. 遵循中的指示佈啟動亞 Amazon EC2 實例（控制台）建執行個體。
2. 啟動執行個體時，請記得在 [新增標籤] 頁面上指定標籤。如需如何指定標籤的詳細資訊，請參閱啟動亞 Amazon EC2 實例（控制台）。

驗證 CodeDeploy 代理程式是否在執行個體上執行

- 按照中確認 CodeDeploy 代理程式正在執行的說明驗證代理程式是否正在執行。

成功佈建執行個體並確認 CodeDeploy 代理程式正在執行之後，請前往下一個步驟。

步驟 5：建立應用程式和部署群組

在此步驟中，您將使用 CodeDeploy 主控台或建立應 AWS CLI 用程式和部署群組，以便從 GitHub 存放庫部署範例修訂版本。

建立應用程式和部署群組 (主控台)

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在瀏覽窗格中，展開 [部署]，然後選擇 [應用程式]。
3. 選擇 Create application (建立應用程式)，然後選取 Custom application (自訂應用程式)。
4. 在 Application name (應用程式名稱) 中，輸入 **`CodeDeployGitHubDemo-App`**。
5. 在 Compute Platform (運算平台) 中，選擇 EC2/On-premises (EC2/ 現場部署)。
6. 選擇建立應用程式。
7. 在 Deployment groups (部署群組) 標籤中，選擇 Create deployment group (建立部署群組)。
8. 在 Deployment group name (部署群組名稱) 中，輸入 **`CodeDeployGitHubDemo-DepGrp`**。
9. 在服務角色中，選擇您在為其建立 CodeDeploy 服務角色中建立的服務角色名稱 CodeDeploy。
10. 在 Deployment type (部署類型) 中，選擇 In-place (就地)。

11. 在環境組態中，根據您使用的執行個體類型，選擇 Amazon EC2 執行個體或現場部署執行個體。對於 Key (金鑰) 和 Value (數值)，輸入套用到您的執行個體的標籤金鑰和數值，做為 步驟 4：佈建執行個體 的一部分。
12. 在 [部署組態] 中，選擇CodeDeployDefault。 AllatOnce。
13. 在 Load Balancer (負載平衡器) 中，清除 Enable load balancing (啟用負載平衡)。
14. 展開 Advanced (進階)。
15. 在 Alarms (警示) 中，選取 Ignore alarm configuration (忽略警示組態)。
16. 選擇 Create deployment group (建立部署群組)，然後繼續進行下一個步驟。

建立應用程式和部署群組 (CLI)

1. 呼叫命create-application令以建立 CodeDeploy 名稱為的應用程式CodeDeployGitHubDemo-App：

```
aws deploy create-application --application-name CodeDeployGitHubDemo-App
```

2. 呼叫 create-deployment-group 命令，建立名為 CodeDeployGitHubDemo-DepGrp 的部署群組。

   - 如果您要部署到 Amazon EC2 執行個體，*ec2 ####*是作為其中一部分套用到 Amazon EC2 執行個體的 Amazon EC2 執行個體標籤金鑰。步驟 4：佈建執行個體
   - 如果您要部署到 Amazon EC2 執行個體，*ec2 ###*是作為其中一部分套用到 Amazon EC2 執行個體的 Amazon EC2 執行個體標籤值。步驟 4：佈建執行個體
   - 如果您要部署到內部部署執行個體，則*on-premises-tag-key*是將內部部署執行個體標記金鑰套用至內部部署執行個體，做為其中的一部分步驟 4：佈建執行個體。
   - 如果您要部署到內部部署執行個體，*on-premises-tag-value*就是將內部部署執行個體標記值套用至內部部署執行個體做為的一部分步驟 4：佈建執行個體。
   - *service-role-arn*是您在建立服務角色中建立之服務角色的服務角色 ARN。CodeDeploy(若要尋找服務角色 ARN，請按照取得服務角色 ARN (CLI) 中的指示)。

```
aws deploy create-deployment-group --application-name CodeDeployGitHubDemo-App
 --ec2-tag-filters Key=ec2-tag-key,Type=KEY_AND_VALUE,Value=ec2-tag-value --on-
premises-tag-filters Key=on-premises-tag-key,Type=KEY_AND_VALUE,Value=on-premises-
tag-value --deployment-group-name CodeDeployGitHubDemo-DepGrp --service-role-
arn service-role-arn
```

> **Note**
>
> 該create-deployment-group命令支援建立觸發器，以便將 Amazon SNS 通知傳送給主題訂閱者有關部署和執行個體中指定事件的主題訂閱者。該命令還支援自動復原部署和設定警示以停止部署的選項，以便在符合 Amazon CloudWatch 警示中的監控閾值時停止部署。這些動作的指令不包括在本教學課程中。

步驟 6：將應用程式部署到執行個體

在此步驟中，您可以使用 CodeDeploy 主控台或 AWS CLI 將範例修訂從 GitHub 存放庫部署到執行個體。

部署修訂版本 (主控台)

1. 在 Deployment group details (部署群組詳細資訊) 頁面上，選擇 Create deployment (建立部署)。
2. 在 Deployment group (部署群組) 中，選擇 **`CodeDeployGitHubDemo-DepGrp`**。
3. 在修訂版本類型中，選擇GitHub。
4. 在 Connect 至中 GitHub，執行下列其中一個動作：
   - 若要為 CodeDeploy 應用程式與 GitHub 帳戶建立連線，請 GitHub 在個別的網頁瀏覽器標籤中登出。在GitHub 帳戶中，輸入識別此連線的名稱，然後選擇 [Connect 線至] GitHub。網頁會提示您授 CodeDeploy 權與名 GitHub 為的應用程式進行互動`CodeDeployGitHubDemo-App`。繼續步驟 5。
   - 若要使用已建立的連線，請在GitHub帳戶中選取其名稱，然後選擇 [Connect 線至] GitHub。繼續步驟 7。
   - 若要建立與其他 GitHub 帳戶的連線，請 GitHub 在個別的網頁瀏覽器標籤中登出。選擇 [Connect 到其他 GitHub帳戶]，然後選擇 [Connect 到] GitHub。繼續步驟 5。
5. 依照「登入」頁面上的指示使用您的 GitHub 帳戶登入。
6. 在 授權應用程式 頁面上，請選擇 授權應用程式。
7. 在「 CodeDeploy 建立部署」頁面的「存放庫名稱」中，輸入您用來登入的使用 GitHub 者名稱，後面接著正斜線 (/)，後面接著您推送應用程式修訂版本的儲存區域名稱 (例如，***`my-github-user-name`*`/CodeDeployGitHubDemo`**)。

   如果您不確定要輸入的值，或者您若想要指定不同的儲存庫：

a. 在單獨的 Web 瀏覽器選項卡中，轉到GitHub 儀表板。

b. 在 Your repositories (您的儲存庫) 中，將滑鼠指標移至目標儲存庫名稱上。工具提示隨即出現，其中顯示 GitHub 使用者或組織名稱，後面接著正斜線 (/)，後面接著存放庫的名稱。輸入這個值到 Repository name (儲存庫名稱)。

> **Note**
>
> 如果目標存放庫名稱未顯示在您的儲存庫中，請使用「搜尋 GitHub」方塊來尋找目標儲存庫以及使用 GitHub 者或組織名稱。

8. 在「確認 ID」方塊中，輸入與推送應用程式修訂至相關聯的提交 ID GitHub。

   如果您不確定要輸入的值：

   a. 在單獨的 Web 瀏覽器選項卡中，轉到GitHub 儀表板。

   b. 在 Your repositories (您的儲存庫) 中，選擇 CodeDeployGitHubDemo。

   c. 在提交列表中，找到與推送應用程序修訂版相關聯的提交 ID 並將其複製到 GitHub。此 ID 通常長度為 40 個字元，並且由字母和數字所組成。(請勿使用較短版本的遞交 ID，其通常是較長版本的前 10 個字元)。

   d. 將遞交 ID 貼至 Commit ID (遞交 ID) 方塊中。

9. 選擇 Deploy (部署)，並繼續下一個步驟。

若要部署修訂版 (CLI)

在您呼叫任何 AWS CLI 與之互動的create-deployment命令 GitHub (例如接下來要呼叫的命令) 之前，您必須 CodeDeploy 授予使用 GitHub 者帳戶與`CodeDeployGitHubDemo-App`應用程式互動 GitHub 的權限。目前，您必須使用 CodeDeploy 控制台來執行此操作。

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在瀏覽窗格中，展開 [部署]，然後選擇 [應用程式]。

3. 選擇 CodeDeployGitHubDemo-App。
4. 在 Deployments (部署) 標籤上，選擇 Create deployment (建立部署)。

> **Note**
>
> 您將無法建立新的部署。這是目前 CodeDeploy 允許代表您的 GitHub 使用者帳戶進 GitHub 行互動的唯一方法。

5. 在部署群組中，選擇 CodeDeployGitHubDemo-DepGrp。
6. 在修訂版本類型中，選擇GitHub。
7. 在 Connect 至中 GitHub，執行下列其中一個動作：

   - 若要為 CodeDeploy 應用程式與 GitHub 帳戶建立連線，請 GitHub 在個別的網頁瀏覽器標籤中登出。在GitHub 帳戶中，輸入識別此連線的名稱，然後選擇 [Connect 線至] GitHub。網頁會提示您授權 CodeDeploy 與名 GitHub 為的應用程式進行互動`CodeDeployGitHubDemo-App`。繼續步驟 8。
   - 若要使用已建立的連線，請在GitHub帳戶中選取其名稱，然後選擇 [Connect 線至] GitHub。繼續步驟 10。
   - 若要建立與其他 GitHub 帳戶的連線，請 GitHub 在個別的網頁瀏覽器標籤中登出。選擇 [Connect 到其他 GitHub帳戶]，然後選擇 [Connect 到] GitHub。繼續步驟 8。

8. 依照「登入」頁面上的指示， GitHub 使用您的使用者名稱或電子郵件和密碼登入。
9. 在 授權應用程式 頁面上，請選擇 授權應用程式。
10. 在 [CodeDeploy 建立部署] 頁面上，選擇 [取消]。
11. 調用create-deployment命令將修訂從 GitHub 存儲庫部署到實例，其中：

    - *`repository`* 是您的 GitHub 帳戶名稱，後面接著正斜線 (/)，後面接著儲存庫的名稱 (`CodeDeployGitHubDemo`)，例如。`MyGitHubUserName/CodeDeployGitHubDemo`

      如果您不確定要使用的值，或者您若想要指定不同的儲存庫：

      1. 在單獨的 Web 瀏覽器選項卡中，轉到GitHub 儀表板。
      2. 在 Your repositories (您的儲存庫) 中，將滑鼠指標移至目標儲存庫名稱上。工具提示隨即出現，其中顯示 GitHub 使用者或組織名稱，後面接著正斜線 (/)，後面接著存放庫的名稱。這是要使用的值。

> **Note**
>
> 如果目標存放庫名稱未出現在您的儲存庫中，請使用 GitHub「搜尋」方塊來尋找目標儲存庫以及對應的使用 GitHub 者或組織名稱。

- *commit-id* 是與您推送至儲存庫 (例如 f835159a...528eb76f) 的應用程式修訂版的版本相關的遞交。

  如果您不確定要使用的值：

  1. 在單獨的 Web 瀏覽器選項卡中，轉到GitHub 儀表板。
  2. 在 Your repositories (您的儲存庫) 中，選擇 CodeDeployGitHubDemo。
  3. 在提交列表中，找到與推送應用程序修訂版本相關聯的提交 ID GitHub。此 ID 通常長度為 40 個字元，並且由字母和數字所組成。(請勿使用較短版本的遞交 ID，其通常是較長版本的前 10 個字元)。使用此值。

如果您使用的是本機 Linux、macOS 或 Unix 電腦：

```
aws deploy create-deployment \
  --application-name CodeDeployGitHubDemo-App \
  --deployment-config-name CodeDeployDefault.OneAtATime \
  --deployment-group-name CodeDeployGitHubDemo-DepGrp \
  --description "My GitHub deployment demo" \
  --github-location repository=repository,commitId=commit-id
```

如果您正在使用本機 Windows 電腦：

```
aws deploy create-deployment --application-name CodeDeployGitHubDemo-App --
deployment-config-name CodeDeployDefault.OneAtATime --deployment-group-name
 CodeDeployGitHubDemo-DepGrp --description "My GitHub deployment demo" --github-
location repository=repository,commitId=commit-id
```

步驟 7：監控和驗證部署

在此步驟中，您將使用 CodeDeploy 主控台或 AWS CLI 來驗證部署是否成功。您將使用 Web 瀏覽器中查看部署到您建立或設定執行個體的網頁。

> **Note**
>
> 如果您要部署到 Ubuntu Server 執行個體，請使用您自己的測試策略來判斷部署的修訂版在執行個體上是否如預期般運作，然後移至下一個步驟。

監控和驗證部署 (主控台)

1. 在瀏覽窗格中，展開 [部署]，然後選擇 [部署]。
2. 在部署清單中，尋找應用程式值為 CodeDeployGitHubDemo-App 且部署群組值為 CodeDeployGitHubDemo-DepGrp 的資料列。如果 Succeeded (成功) 或 Failed (失敗) 未出現在 Status (狀態) 欄中，請定期選擇 Refresh (重新整理) 按鈕。
3. 如果 Failed (失敗) 出現在 Status (狀態) 欄中，請按照 查看實例詳細信息（控制台） 中的指示，排除部署問題。
4. 如果 Succeeded (成功) 出現在 Status (狀態) 欄中，您現在可以透過 Web 瀏覽器驗證部署。我們的範例修訂版將單一網頁部署到執行個體。如果您要部署到 Amazon EC2 執行個體，請在網頁瀏覽器中移至`http://`*`public-dns`*執行個體 (例如`http://ec2-01-234-567-890.compute-1.amazonaws.com`)。
5. 如果您可以看到網頁，那麼恭喜！現在您已成功使用 AWS CodeDeploy 來部署修訂版 GitHub，您可以跳到步驟 8：清除.

若要監控和驗證部署 (CLI)

1. 呼叫 list-deployments 命令以取得名為 `CodeDeployGitHubDemo-App` 之應用程式的部署 ID 以及名為 `CodeDeployGitHubDemo-DepGrp` 的部署群組。

```
aws deploy list-deployments --application-name CodeDeployGitHubDemo-App --
deployment-group-name CodeDeployGitHubDemo-DepGrp --query "deployments" --output
 text
```

2. 呼叫 get-deployment 命令，提供從 list-deployments 命令輸出的部署 ID：

```
aws deploy get-deployment --deployment-id deployment-id --query "deploymentInfo.
[status, creator]" --output text
```

3. 如果傳回 Failed (失敗) ，請按照 查看實例詳細信息（控制台） 中的指示來排除部署問題。

4. 如果傳回 Succeeded (成功)，您現在可以嘗試透過 Web 瀏覽器驗證部署。我們的範例修訂版是部署到執行個體的單一網頁。如果您要部署到 Amazon EC2 執行個體，可以前往 `http://public-dns` Amazon EC2 執行個體 (例如`http://ec2-01-234-567-890.compute-1.amazonaws.com`)，在網頁瀏覽器中檢視此頁面。
5. 如果您可以看到網頁，那麼恭喜！您已成功使 AWS CodeDeploy 用從 GitHub 儲存庫部署。

步驟 8：清除

若要避免在本教學中使用的資源進一步收費，您必須終止 Amazon EC2 執行個體及其相關資源。或者，您可以刪除與此自學課程相關聯的 CodeDeploy部署元件記錄。如果您僅在本教程中使用 GitHub 存儲庫，則現在也可以將其刪除。

若要刪除 AWS CloudFormation 堆疊 (如果您使用 AWS CloudFormation 範本建立 Amazon EC2 執行個體)

1. 請登入 AWS Management Console 並開啟 AWS CloudFormation 主控台，網址為 https://console.aws.amazon.com/cloudformation。
2. 在 Stacks (堆疊) 欄中，選擇開頭為 `CodeDeploySampleStack` 的堆疊。
3. 選擇刪除。
4. 出現提示時，選擇 Delete stack (刪除堆疊)。Amazon EC2 執行個體以及相關的 IAM 執行個體設定檔和服務角色都會遭到刪除。

若要手動取消註冊和清除內部部署執行個體 (如果您佈建的是內部部署執行個體)

1. 使用針對此處所代表的內部部署執行個體呼叫取消註冊命令， AWS CLI 以*your-instance-name*及#所在地區的關聯區域：

```
aws deploy deregister --instance-name your-instance-name --no-delete-iam-user --
region your-region
```

2. 從內部部署執行個體呼叫解除安裝命令：

```
aws deploy uninstall
```

手動終止 Amazon EC2 執行個體 (如果您手動啟動 Amazon EC2 執行個體)

1. 登入 AWS Management Console 並開啟 Amazon EC2 主控台，網址為 https://console.aws.amazon.com/ec2/。
2. 在導覽窗格的 Instances (執行個體) 下方，選擇 Instances (執行個體)。
3. 選取您要終止之 Amazon EC2 執行個體旁邊的核取方塊。在 Actions (動作) 選單中，指向 Instance State (執行個體狀態)，然後選擇 Terminate (終止)。
4. 出現提示時，選擇 Yes, Terminate (是，終止)。

刪除 CodeDeploy 部署元件記錄

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在瀏覽窗格中，展開 [部署]，然後選擇 [應用程式]。
3. 選擇 CodeDeployGitHubDemo-App。
4. 選擇刪除應用程式。
5. 當出現提示時，輸入 **`Delete`**，然後選擇 Delete (刪除)。

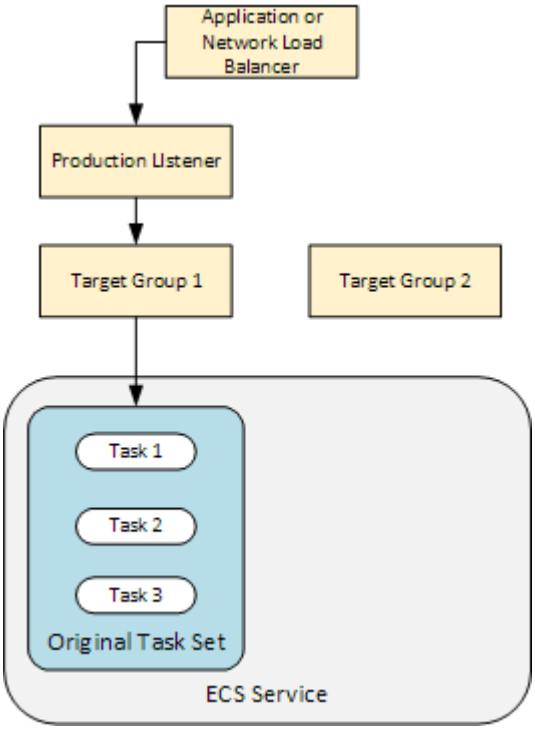
若要刪除您的 GitHub儲存庫

請參閱GitHub 說明中的刪除存放庫。

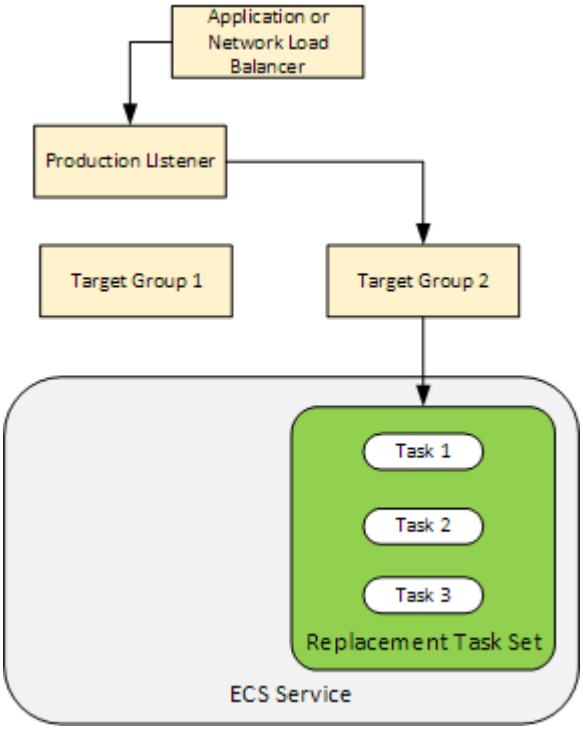
教學課程：將應用程式部署到 Amazon ECS

在本教學中，您將學習如何使用將應用 CodeDeploy程式部署到 Amazon ECS。您可以從已建立並部署到 Amazon ECS 的應用程式開始。第一步是加上新標籤修改應用程式的任務定義檔案，以更新您的應用程式。接下來，您可 CodeDeploy 以用來部署更新。在部署期間，將您的更新 CodeDeploy 安裝到新的取代任務集中。然後，它會將生產流量從原始任務集中的 Amazon ECS 應用程式的原始版本轉移到取代任務集中的更新版本。

在 Amazon ECS 部署期間， CodeDeploy 使用設定了兩個目標群組和一個生產流量接聽程式的負載平衡器。下圖顯示在部署開始之前，負載平衡器、生產接聽程式、目標群組和 Amazon ECS 應用程式的關聯性。本教學課程會使用 Application Load Balancer。您也可以使用 Network Load Balancer。



成功部署後，生產流量接聽程式會將流量轉送至新的替換任務集，並終止原始任務集。下圖顯示您的資源在成功部署後如何相關。如需詳細資訊，請參閱 Amazon ECS 部署期間會發生什麼情況。



如需如何使用將應用程式部署 AWS CLI 到 Amazon ECS 的相關資訊，請參閱教學課程：使用藍/綠部署建立服務。如需如何使用 CodePipeline 來偵測 Amazon ECS 服務並自動部署變更的相關資訊 CodeDeploy，請參閱教學課程：使用 Amazon ECR 來源建立管道和 ECS 到部署。CodeDeploy

完成此自學課程後，您可以使用您建立的 CodeDeploy 應用程式和部署群組，在中新增部署驗證測試教學課程：透過驗證測試部署 Amazon ECS 服務。

主題

必要條件

若要完成此教學課程，您必須先：

- 完成 開始使用 CodeDeploy中的步驟 2 和 3。

- 建立設定兩個目標群組和一個監聽器的「Application Load Balancer」。如需有關使用主控台建立負載平衡器的資訊，請參閱為 CodeDeploy Amazon ECS 部署設定負載平衡器、目標群組和接聽程式。如需使用建立負載平衡器的相關資訊 AWS CLI，請參閱 Amazon 彈性容器服務使用者指南中的步驟 1：建立 Application Load Balancer 器。當您建立負載平衡器時，請為此教學課程記下以下事項：
  - 負載平衡器的名稱。
  - 目標群組的名稱。
  - 負載平衡器接聽程式所使用的連接埠。
- 建立 Amazon ECS 叢集和服務。如需詳細資訊，請參閱 Amazon 彈性容器服務使用者指南中教學課程：使用藍/綠部署建立服務中的步驟 2、3 和 4。請為此教學課程記下以下事項：
  - 您的 Amazon ECS 群集的名稱。
  - Amazon ECS 服務所使用之任務定義的 ARN。
  - Amazon ECS 服務所使用的容器名稱。
- 為您的 AppSpec 檔案建立一個 Amazon S3 儲存貯體。

步驟 1：更新您的 Amazon ECS 應用程序

在本節中，您將使用其任務定義的新修訂版更新 Amazon ECS 應用程式。更新後的修訂會新增金鑰和標籤對。在中 步驟 3：使用主 CodeDeploy 控台部署應用程式，您可以部署 Amazon ECS 應用程式的更新版本。

更新您的任務定義

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇 Task Definitions (任務定義)。
3. 選擇 Amazon ECS 服務使用的任務定義。
4. 選擇任務定義修訂版，然後選擇建立新修訂版、建立新修訂版。
5. 針對本教學課程，稍微更新任務定義，只新增一個標籤。在頁面底部的「標籤」下，輸入新的金鑰和值配對，以建立新標籤。
6. 選擇建立。

   您的任務定義的修訂版本號碼會以 1 遞增。
7. 選擇 JSON 標籤。請記下以下項目，因為您在下一個步驟中需要此資訊。

- taskDefinitionArn 的值。格式為 arn:aws:ecs:*aws-region*:*account-id*:task-definition/*task-definition-family*:*task-definition-revision*。這是已更新之任務定義的 ARN。
- 在 containerDefinitions 元素中，name 的值。這是容器的名稱。
- 在 portMappings 元素中，containerPort 的值。這是容器的連接埠。

步驟 2：建立檔 AppSpec 案

在本節中，您可以建立 AppSpec 檔案並將其上傳到您在本必要條件節中建立的 Amazon S3 儲存貯體。Amazon ECS 部署的 AppSpec 檔案會指定您的任務定義、容器名稱和容器連接埠。如需詳細資訊，請參閱 AppSpec Amazon ECS 部署的檔案範例 及 AppSpec Amazon ECS 部署的「資源」部分。

若要建立您的 AppSpec 檔案

1. 如果您想要使用 YAML 建立 AppSpec 檔案，請建立名為appspec.yml. 如果您想要使用 JSON 建立 AppSpec 檔案，請建立名為appspec.json.
2. 根據您是否為檔案使用 YAML 或 JSON，並將其內容複製到您剛建立的 AppSpec 檔案中，選擇適當的 AppSpec 索引標籤。對於 TaskDefinition 屬性，請使用您在 步驟 1：更新您的 Amazon ECS 應用程序一節記下的任務定義 ARN。

   JSON AppSpec

```
{
  "version": 0.0,
  "Resources": [
    {
      "TargetService": {
        "Type": "AWS::ECS::Service",
        "Properties": {
          "TaskDefinition": "arn:aws:ecs:aws-region-id:aws-account-id:task-
definition/ecs-demo-task-definition:revision-number",
          "LoadBalancerInfo": {
            "ContainerName": "your-container-name",
            "ContainerPort": your-container-port
          }
        }
      }
    }
```

```
  ]
}
```

YAML AppSpec

```
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "arn:aws:ecs:aws-region-id:aws-account-id:task-
definition/ecs-demo-task-definition:revision-number"
        LoadBalancerInfo:
          ContainerName: "your-container-name"
          ContainerPort: your-container-port
```

> **Note**
>
> 您的替換任務集會從原始任務集繼承子網路、安全群組、平台版本及指派的公有 IP 值。您可以在 AppSpec 檔案中設定取代工作集的可選性質，以取代這些值。如需詳細資訊，請參閱 AppSpec Amazon ECS 部署的「資源」部分 及 AppSpec Amazon ECS 部署的檔案範例 。

3. 將 AppSpec 檔案上傳到您建立的 S3 儲存貯體，做為本教學的先決條件。

步驟 3：使用主 CodeDeploy 控台部署應用程式

在本節中，您會建立 CodeDeploy 應用程式和部署群組，以將更新的應用程式部署到 Amazon ECS。在部署期間，將應用程式的生產流量 CodeDeploy 轉移到新的取代工作集中的新版本。若要完成此步驟，您需要下列項目：

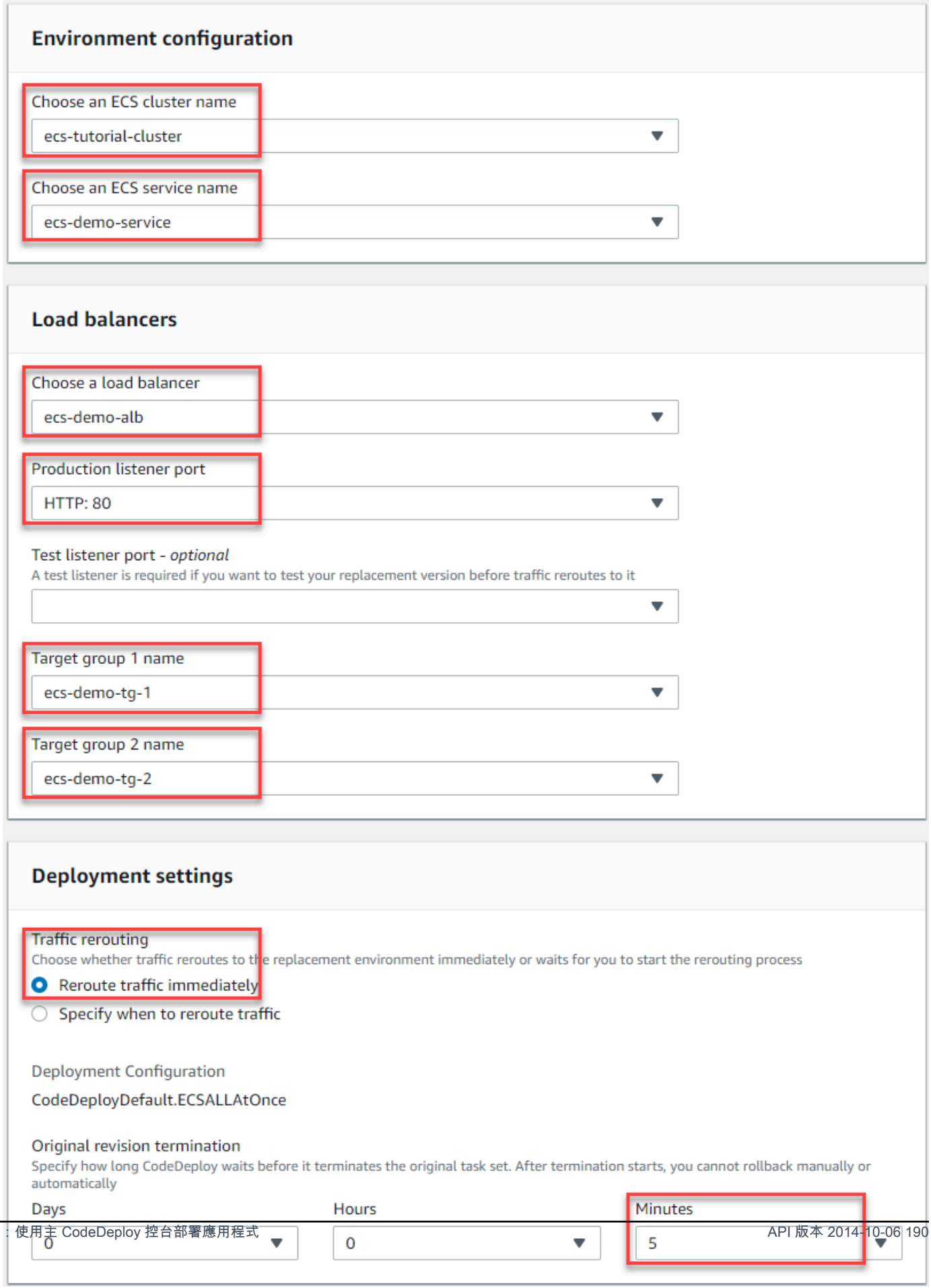
- 您的 Amazon ECS 叢集名稱。
- 您的 Amazon ECS 服務名稱。
- 應用程式負載平衡器名稱。
- 您的生產接聽程式連接埠。
- 您的目標群組名稱。
- 您建立的 S3 儲存貯體的名稱。

若要建立 CodeDeploy 應用程式

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy/。
2. 選擇建立應用程式。
3. 在 Application name (應用程式名稱) 中，輸入 **ecs-demo-codedeploy-app**。
4. 在 Compute Platform (運算平台) 中，選擇 Amazon ECS。
5. 選擇建立應用程式。

若要建立 CodeDeploy 部署群組

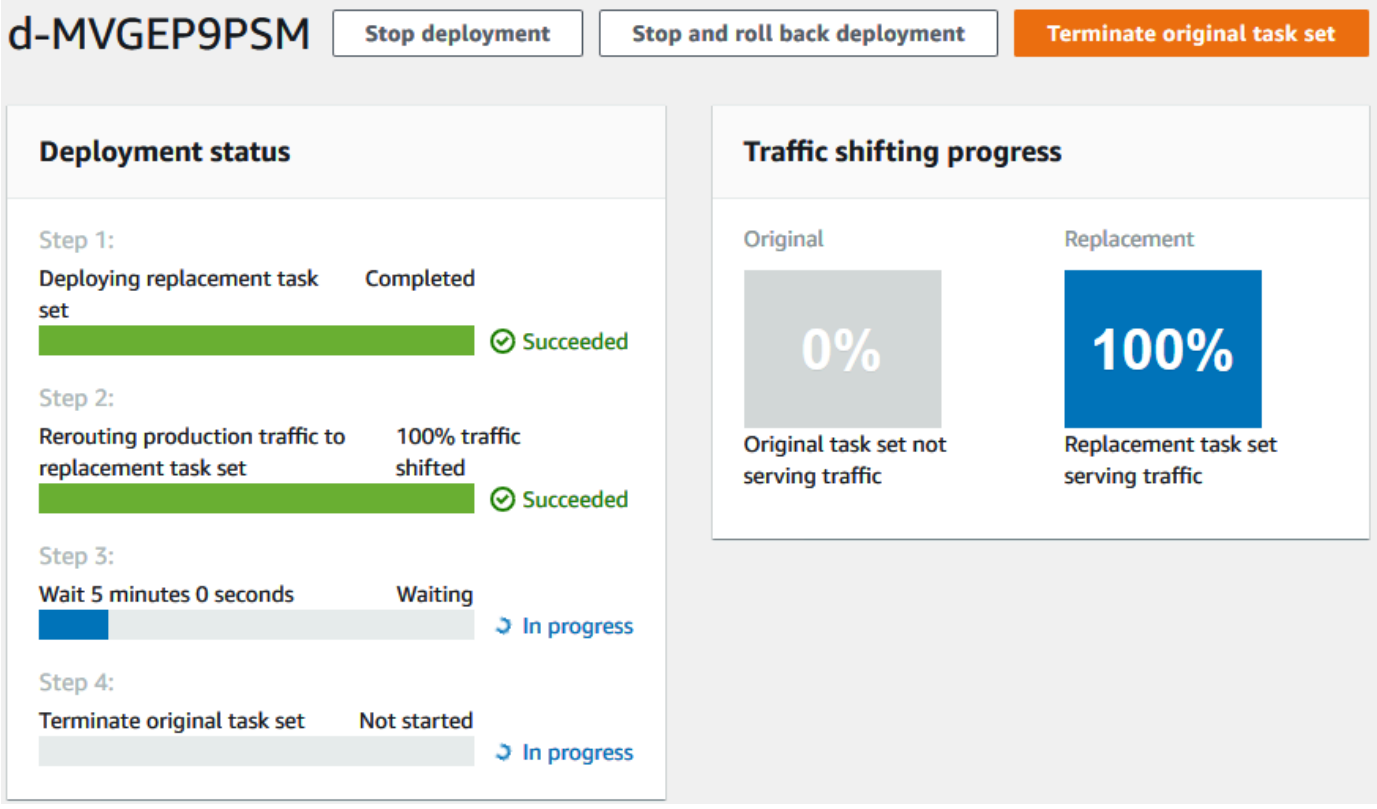
1. 在應用程式頁面的 Deployment groups (部署群組) 標籤上，選擇 Create deployment group (建立部署群組)。
2. 在 Deployment group name (部署群組名稱) 中，輸入 **ecs-demo-dg**。
3. 在服務角色中，選擇授與 Amazon ECS CodeDeploy 存取權的服務角色。如需詳細資訊，請參閱適用於 AWS CodeDeploy的 Identity and Access Management。
4. 在環境組態中，選擇您的 Amazon ECS 叢集名稱和服務名稱。
5. 從負載平衡器中，選擇負載平衡器的名稱，該平衡器為 Amazon ECS 服務提供流量。
6. 在生產接聽程式連接埠中，選擇將生產交易提供給 Amazon ECS 服務的接聽程式的連接埠和通訊協定 (例如 HTTP: 80)。本教學課程不包含選用的測試接聽程式，因此請勿從 Test listener port (測試接聽程式連接埠) 中選擇連接埠。
7. 從 Target group 1 name (目標群組 1 名稱) 和 Target group 2 name (目標群組 2 名稱) 中，選擇在部署期間路由流量的目標群組。確定這些是您為負載平衡器建立的目標群組。何者用於目標群組 1，何者用於目標群組 2，都沒關係。
8. 選擇 Reroute traffic immediately (立即重新路由流量)。
9. 對於 Original revision termination (原始修訂終止)，選擇 0 天、0 小時和 5 分鐘。這可讓您看到部署比使用預設值 (1 小時) 更快完成。


Environment configuration
Choose an ECS cluster name
ecs-tutorial-cluster
Choose an ECS service name
ecs-demo-service
Load balancers
Choose a load balancer
ecs-demo-alb
Production listener port
HTTP: 80
Test listener port - optional
A test listener is required if you want to test your replacement version before traffic reroutes to it
Target group 1 name
ecs-demo-tg-1
Target group 2 name
ecs-demo-tg-2
Deployment settings
Traffic rerouting
Choose whether traffic reroutes to the replacement environment immediately or waits for you to start the rerouting process
Reroute traffic immediately
Specify when to reroute traffic
Deployment Configuration
CodeDeployDefault.ECSALLAtOnce
Original revision termination
Specify how long CodeDeploy waits before it terminates the original task set. After termination starts, you cannot rollback manually or automatically
Days
0
Hours
0
Minutes
5

10. 選擇 Create deployment group (建立部署群組)。

若要部署您的 Amazon ECS 應用程式

1. 從您的部署群組主控台頁面，選擇 Create deployment (建立部署)。
2. 對於部署群組，請選擇ecs-demo-dg。
3. 針對 Revision type (修訂版類型)，選擇 My application is stored in Amazon S3 (我的應用程式存放在 Amazon S3)。在 Revision location (修訂版位置) 中，輸入您的 S3 儲存貯體的名稱。
4. 針對 Revision file type (修訂檔案類型)，視需要選擇 .json 或 .yaml。
5. (選用) 在 Deployment description (部署描述) 中，輸入部署的描述。
6. 選擇 Create deployment (建立部署)。
7. 在 Deployment status (部署狀態) 中，您可以監控部署。100% 的生產流量路由至取代工作集之後，在五分鐘的等待時間到期之前，您可以選擇「終止原始工作集」，立即終止原始工作集。如果您不選擇 Terminate original task set (終止原始任務集)，原始任務集會在您指定的五分鐘等待時間到期之後終止。



步驟 4：清理

下一個自學課程建立在本自學課程之上教學課程：透過驗證測試部署 Amazon ECS 服務，並使用您建立的 CodeDeploy 應用程式和部署群組。如果您想要遵循該教學課程中的步驟，請略過此步驟，並且不要刪除您建立的資源。

> **Note**
>
> 您的 AWS 帳戶不會對您建立的 CodeDeploy 資源產生費用。

這些步驟中的資源名稱是本教學課程中建議的名稱 (例如，**ecs-demo-codedeploy-app**您的 CodeDeploy 應用程式名稱)。如果您使用不同的名稱，請務必在清理期間使用這些名稱。

1. 使用delete-deployment-group命令刪除部 CodeDeploy 署群組。

```
aws deploy delete-deployment-group --application-name ecs-demo-codedeploy-app --
deployment-group-name ecs-demo-dg --region aws-region-id
```

2. 使用刪除應用程式命令刪除應用程式。 CodeDeploy

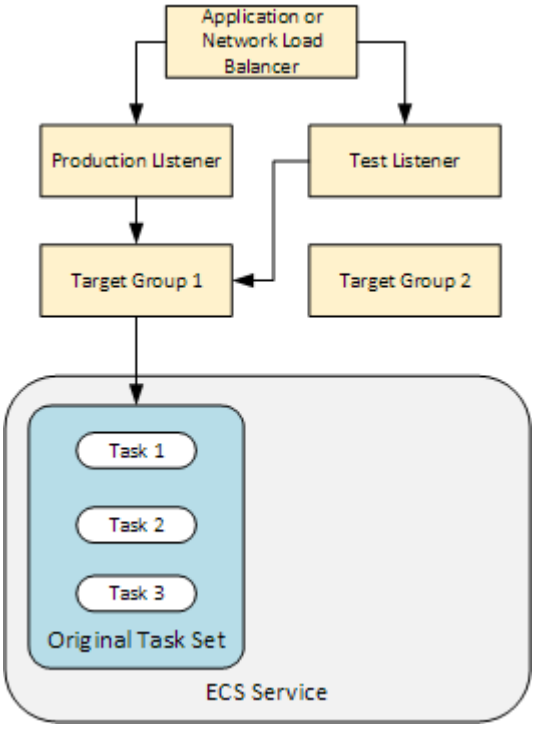
```
aws deploy delete-application --application-name ecs-demo-codedeploy-app --
region aws-region-id
```

教學課程：透過驗證測試部署 Amazon ECS 服務

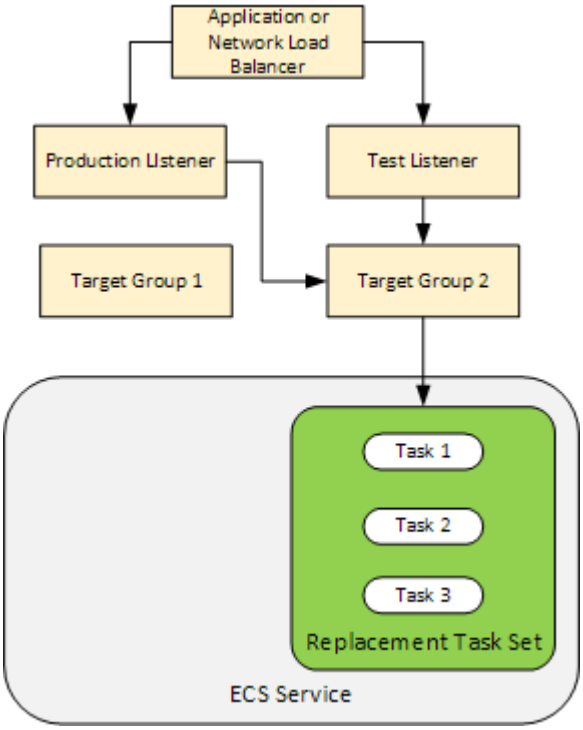
在本教學中，您將學習如何使用 Lambda 函數來驗證部分更新的 Amazon ECS 應用程式部署。本教學課程使用您在教學課程：將應用程式部署到 Amazon ECS其中使用的 CodeDeploy 應用程式、CodeDeploy 部署群組和 Amazon ECS 應用程式。請先完成該教學課程，再開始本教學課程。

若要新增驗證測試，請先在 Lambda 函數中實作測試。接下來，在部署 AppSpec 檔案中，為要測試的生命週期勾點指定 Lambda 函數。如果驗證測試失敗，部署會停止、轉返，並標記為失敗。如果測試成功，部署會繼續下一個部署生命週期事件或勾點。

在使用驗證測試進行 Amazon ECS 部署期間， CodeDeploy 使用設定了兩個目標群組的負載平衡器：一個生產流量接聽程式和一個測試流量接聽程式。下圖顯示在部署開始之前，負載平衡器、生產和測試接聽程式、目標群組以及 Amazon ECS 應用程式的關聯性。本教學課程會使用 Application Load Balancer。您也可以使用 Network Load Balancer。



在 Amazon ECS 部署期間，有五個生命週期掛鉤可供測試。本教學課程會在第三個生命週期部署勾點 `AfterAllowTestTraffic` 期間實作一個測試。如需詳細資訊，請參閱 Amazon ECS 部署的生命週期事件掛鉤清單。成功部署後，生產流量接聽程式會將流量轉送至新的替換任務集，並終止原始任務集。下圖顯示您的資源在成功部署後如何相關。如需詳細資訊，請參閱 Amazon ECS 部署期間會發生什麼情況。



> **Note**
>
> 完成本教學課程可能會向您的 AWS 帳戶收取費用。這些費用包括 CodeDeploy AWS Lambda、和的可能費用 CloudWatch。如需詳細資訊，請參閱AWS Lambda 定AWS CodeDeploy 價、定價和 Amazon CloudWatch 定價。

主題

必要條件

若要成功完成此教學課程，您必須先：

- 完成中的先決條必要條件件教學課程：將應用程式部署到 Amazon ECS。
- 完成「教學課程：將應用程式部署到 Amazon ECS」中的步驟。記下以下項目：
  - 負載平衡器的名稱。
  - 目標群組的名稱。
  - 負載平衡器接聽程式所使用的連接埠。
  - 負載平衡器的 ARN。這可用來建立新的接聽程式。
  - 其中一個目標群組的 ARN。這可用來建立新的接聽程式。
  - 您建立的 CodeDeploy 應用程式和部署群組。
  - 您建立的 CodeDeploy 部署所使用的 AppSpec 檔案。您在此教學課程中會編輯此檔案。

步驟 1：建立測試接聽程式

具有驗證測試的 Amazon ECS 部署需要第二個接聽程式。此接聽程式可用來為替代任務集中更新的 Amazon ECS 應用程式提供測試流量。您的驗證測試會針對測試流量執行。

測試流量的接聽程式可以使用任一個目標群組。使用建立偵聽程式 AWS CLI 命令建立第二個接聽程式，其預設規則會將測試流量轉送至連接埠 8080。使用負載平衡器的 ARN 和其中一個目標群組的 ARN。

```
aws elbv2 create-listener --load-balancer-arn your-load-balancer-arn \
--protocol HTTP --port 8080 \
--default-actions Type=forward,TargetGroupArn=your-target-group-arn --region your-aws-region
```

步驟 2：更新您的 Amazon ECS 應用程序

在本節中，您將更新 Amazon ECS 應用程式以使用其任務定義的新修訂版本。您建立新的修訂版，新增一個標籤稍做更新。

更新您的任務定義

1. 開啟 Amazon ECS 傳統主控台，網址為 https://console.aws.amazon.com/ecs/。
2. 在導覽窗格中，選擇 Task Definitions (任務定義)。
3. 選取 Amazon ECS 服務所使用之任務定義的核取方塊。
4. 選擇 Create new revision (建立新的修訂)。
5. 稍微更新任務定義，只新增一個標籤。在頁面底部的 Tags (標籤) 中，輸入新的鍵值對來建立新的標籤。
6. 選擇建立。您應該會看到任務定義的修訂版編號已增加 1。
7. 選擇 JSON 標籤。記下 `taskDefinitionArn` 的值。格式為 `arn:aws:ecs:`*aws-region*`:`*account-id*`:task-definition/`*task-definition-family*`:` *task-definition-revision*。這是已更新之任務定義的 ARN。

步驟 3：建立生命週期勾點 Lambda 函數

在本節中，您將為您的 Amazon ECS 部署`AfterAllowTestTraffic`勾點實作一個 Lambda 函數。Lambda 函數會在安裝更新的 Amazon ECS 應用程式之前執行驗證測試。在本教學課程中，Lambda 函數會傳回`Succeeded`。在真實世界部署期間，驗證測試會傳回 `Succeeded` 或

`Failed`，取決於驗證測試的結果。此外，在真實世界部署期間，您可以為一或多個其他 Amazon ECS 部署生命週期事件掛鉤 (`BeforeInstall`、`AfterInstallBeforeAllowTraffic`、和`AfterAllowTraffic`) 實作 Lambda 測試函數。如需詳細資訊，請參閱 Amazon ECS 部署的生命週期事件掛鉤清單。

若要建立 Lambda 函數，您需要具備 IAM 角色。此角色授與 Lambda 函數寫入 CloudWatch 記錄和設定 CodeDeploy 生命週期掛接狀態的權限。

若要建立一個 IAM 角色

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 從導覽窗格，選擇 Roles (角色)，然後選擇 Create role (建立角色)。
3. 建立具備下列屬性的角色：

   - Trusted entity (信任實體)：AWS Lambda。
   - 權限:AWSLambdaBasicExecutionRole. 這會授予您的 Lambda 函數寫入 CloudWatch 記錄的權限。
   - Role name (角色名稱)：**`lambda-cli-hook-role`**。

   如需詳細資訊，請參閱建立 AWS Lambda 執行角色。
4. 將許可 `codedeploy:PutLifecycleEventHookExecutionStatus` 連接至您建立的角色。這會授予 Lambda 函數在部署期間設定 CodeDeploy 生命週期掛接狀態的權限。如需詳細資訊，請參閱AWS Identity and Access Management 使用指南和 CodeDeploy API 參考PutLifecycleEventHookExecutionStatus中的新增 IAM 身分許可。

若要建立**`AfterAllowTestTraffic`**掛接 Lambda 函數

1. 使用下列內容建立名為 `AfterAllowTestTraffic.js` 的檔案。

```
'use strict';

 const AWS = require('aws-sdk');
 const codedeploy = new AWS.CodeDeploy({apiVersion: '2014-10-06'});

 exports.handler = (event, context, callback) => {

  console.log("Entering AfterAllowTestTraffic hook.");
```

```
 // Read the DeploymentId and LifecycleEventHookExecutionId from the event payload
 var deploymentId = event.DeploymentId;
 var lifecycleEventHookExecutionId = event.LifecycleEventHookExecutionId;
 var validationTestResult = "Failed";

 // Perform AfterAllowTestTraffic validation tests here. Set the test result
 // to "Succeeded" for this tutorial.
 console.log("This is where AfterAllowTestTraffic validation tests happen.")
 validationTestResult = "Succeeded";

 // Complete the AfterAllowTestTraffic hook by sending CodeDeploy the validation
status
 var params = {
  deploymentId: deploymentId,
  lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
  status: validationTestResult // status can be 'Succeeded' or 'Failed'
 };

 // Pass CodeDeploy the prepared validation test results.
 codedeploy.putLifecycleEventHookExecutionStatus(params, function(err, data) {
  if (err) {
   // Validation failed.
   console.log('AfterAllowTestTraffic validation tests failed');
   console.log(err, err.stack);
   callback("CodeDeploy Status update failed");
  } else {
   // Validation succeeded.
   console.log("AfterAllowTestTraffic validation tests succeeded");
   callback(null, "AfterAllowTestTraffic validation tests succeeded");
  }
 });
}
```

2. 建立 Lambda 部署套件。

```
zip AfterAllowTestTraffic.zip AfterAllowTestTraffic.js
```

3. 使用此`create-function`命令為您的`AfterAllowTestTraffic`掛接建立 Lambda 函數。

```
aws lambda create-function --function-name AfterAllowTestTraffic \
       --zip-file fileb://AfterAllowTestTraffic.zip \
       --handler AfterAllowTestTraffic.handler \
```

```
    --runtime nodejs10.x \
    --role arn:aws:iam::aws-account-id:role/lambda-cli-hook-role
```

4. 在create-function回應中記下您的 Lambda 函數 ARN。當您在下一個步驟中更新 CodeDeploy 部署的 AppSpec 檔案時，請使用此 ARN。

步驟 4：更新檔 AppSpec案

在本節中，您將使用Hooks區段更新 AppSpec 檔案。在Hooks本節中，您可以為AfterAllowTestTraffic生命週期勾點指定 Lambda 函數。

若要更新您的 AppSpec 檔案

1. 開啟您在中建立 步驟 2：建立檔 AppSpec 案 的 AppSpec 檔案檔案教學課程：將應用程式部署到 Amazon ECS。
2. 使用您在 步驟 2：更新您的 Amazon ECS 應用程序中記下的任務定義 ARN 來更新 TaskDefinition 屬性。
3. 將Hooks區段複製並貼到 AppSpec 檔案檔案中。在AfterAllowTestTraffic使用您在中記錄的 Lambda 函數的 ARN 之後更新 ARN。步驟 3：建立生命週期勾點 Lambda 函數

   JSON AppSpec

```
{
  "version": 0.0,
  "Resources": [
    {
      "TargetService": {
        "Type": "AWS::ECS::Service",
        "Properties": {
          "TaskDefinition": "arn:aws:ecs:aws-region-id:aws-account-id::task-
definition/ecs-demo-task-definition:revision-number",
          "LoadBalancerInfo": {
            "ContainerName": "sample-website",
            "ContainerPort": 80
          }
        }
      }
    }
  ],
  "Hooks": [
```

```
    {
      "AfterAllowTestTraffic": "arn:aws:lambda:aws-region-id:aws-account-
id:function:AfterAllowTestTraffic"
    }
  ]
}
```

YAML AppSpec

```
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "arn:aws:ecs:aws-region-id:aws-account-id::task-
definition/ecs-demo-task-definition:revision-number"
        LoadBalancerInfo:
          ContainerName: "sample-website"
          ContainerPort: 80
Hooks:
  - AfterAllowTestTraffic: "arn:aws:lambda:aws-region-id:aws-account-
id:function:AfterAllowTestTraffic"
```

4. 儲存 AppSpec 檔案並上傳至其 S3 儲存貯體。

步驟 5：使用主 CodeDeploy 控台部署您的 Amazon ECS 服務

在本節中，您會指定測試接聽程式的連接埠，以更新部署群組。這是您在 步驟 1：建立測試接聽程式 中建立的接聽程式。在部署期 CodeDeploy間，請使用測試接聽程式提供給替代工作集的測試流量，在AfterAllowTestTraffic部署生命週期掛接期間執行驗證測試。您的驗證測試會傳回結果 Succeeded，因此部署會繼續進行下一個部署生命週期事件。在真實世界案例中，您的測試函數會傳回 Succeeded 或 Failed。

將測試接聽程式新增至您的部署群組

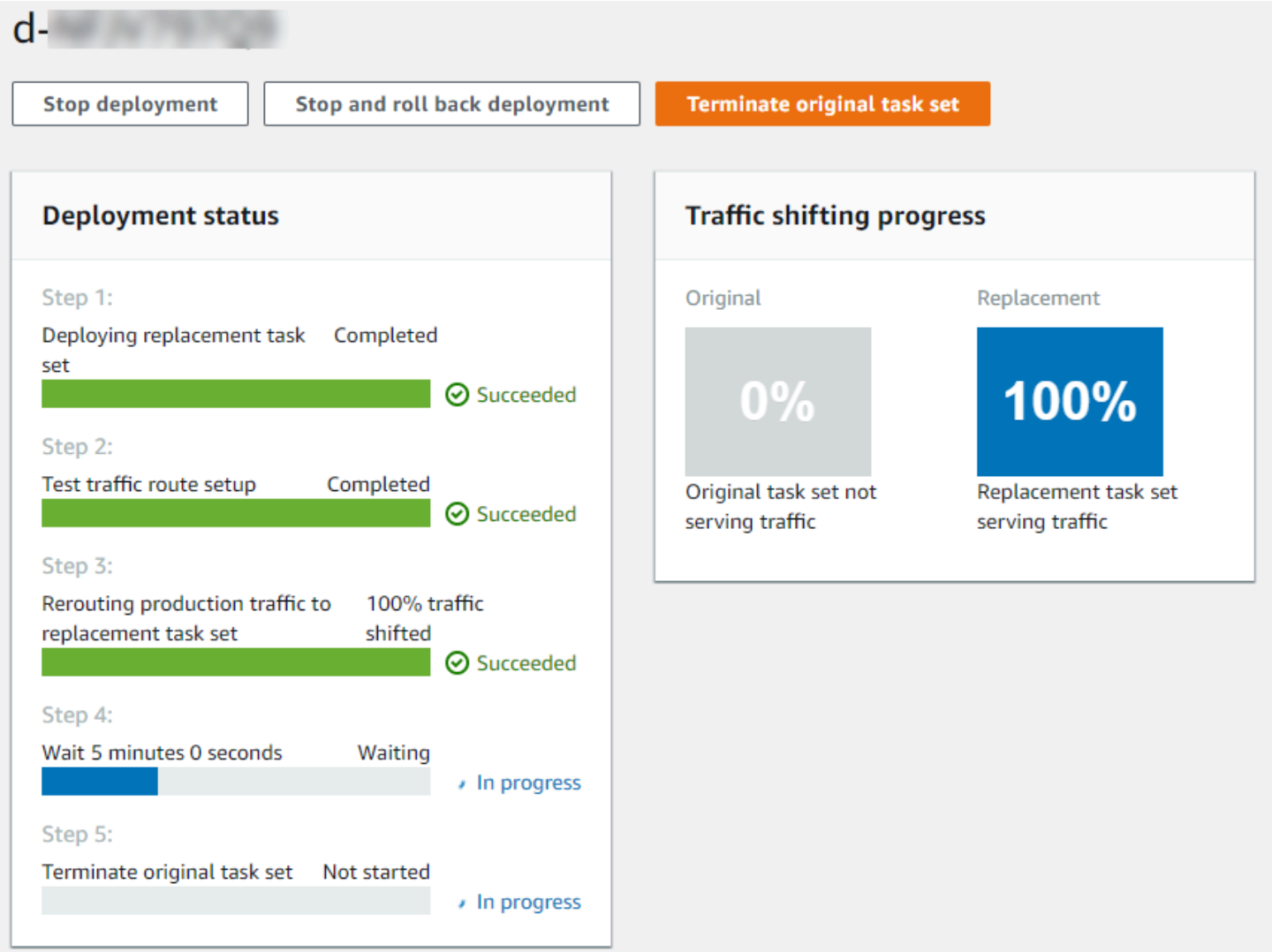
1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy/。
2. 從導覽窗格中，選擇 Applications (應用程式)。
3. 選擇您在教學課程：將應用程式部署到 Amazon ECS中建立的應用程式。如果您使用建議的名稱，則為ecs-demo-codedeploy-app。

4. 在 Deployment group (部署群組) 中，選擇您在教學課程：將應用程式部署到 Amazon ECS中建立的部署群組。如果您使用建議的名稱，則為ecs-demo-dg。
5. 選擇編輯。
6. 從 Test listener port (測試接聽程式連接埠) 中，為您稍早在此教學課程中建立的測試接聽程式，選擇連接埠和通訊協定。此應為 HTTP: 8080。
7. 選擇儲存變更。

若要部署您的 Amazon ECS 應用程式

1. 從您的部署群組主控台頁面，選擇 Create deployment (建立部署)。
2. 對於部署群組，請選擇ecs-demo-dg。
3. 針對 Revision type (修訂版類型)，選擇 My application is stored in Amazon S3 (我的應用程式存放在 Amazon S3)。在修訂位置中，輸入 S3 儲存貯體和 AppSpec 檔案的名稱 (例如，**`s3://my-s3-bucket/appspec.json`**)。
4. 針對 Revision file type (修訂檔案類型)，視需要選擇 .json 或 .yaml。
5. (選用) 在 Deployment description (部署描述) 中，輸入部署的描述。
6. 選擇 Create deployment (建立部署)。

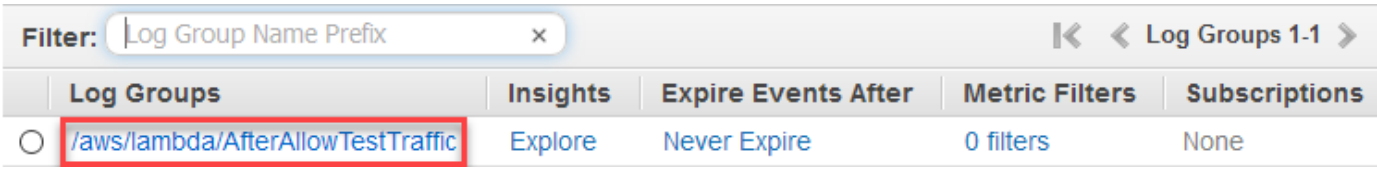
您可以在 Deployment status (部署狀態)中監控部署。將 100% 的生產流量路由至取代任務集之後，您可以選擇「終止原始作業集」以立即終止原始作業集。如果您不選擇 Terminate original task set (終止原始任務集)，原始任務集會在您建立部署群組時指定的持續時間後終止。



步驟 6：在 CloudWatch 日誌中檢視 Lambda 掛接函數輸出

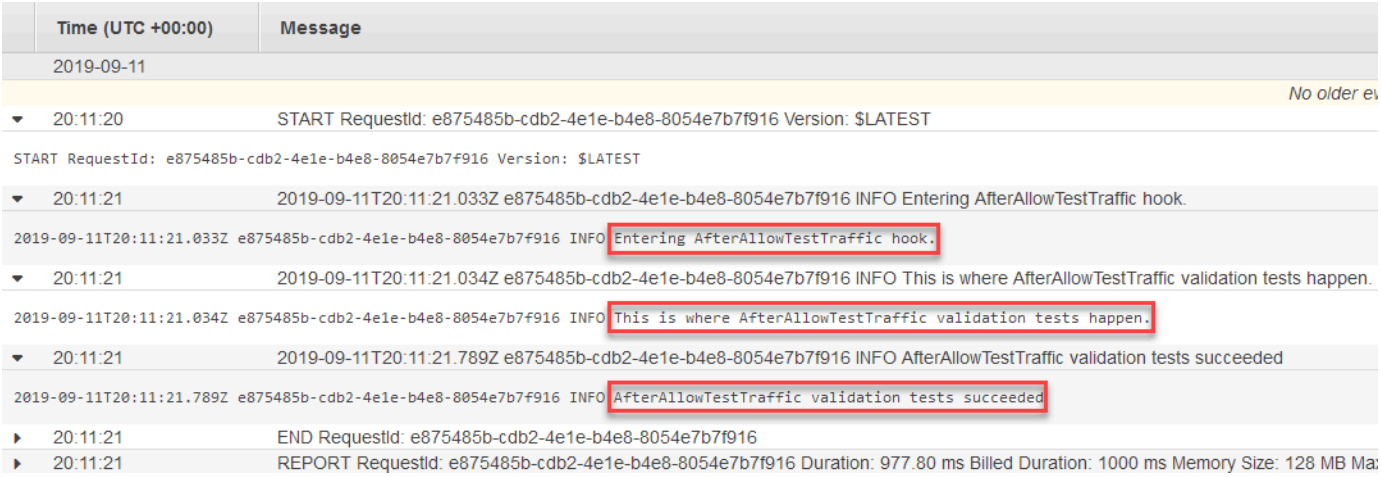
如果 CodeDeploy 部署成功，Lambda 掛接功能中的驗證測試也會成功。您可以通過查看日誌中鉤子功能的 CloudWatch 日誌來確認這一點。

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 從導覽窗格中，選擇 Logs (日誌)。您應該會看到您在 AppSpec 檔案中指定的 Lambda 掛接函數的新記錄群組。



3. 選擇新的日誌群組。這應該是 /aws/羊肉/ AfterAllowTestTrafficHook。

4. 選擇日誌串流。如果您看到多個日誌串流，請選擇在 Last Event Time (上次事件時間) 下具有最新日期和時間的日誌串流。
5. 展開記錄串流事件，以確認 Lambda 掛接函數將成功訊息寫入記錄檔。下面顯示了 `AfterAllowTraffic` Lambda 掛鉤函數是否成功。



步驟 7：清除

完成此教學課程時，清除與其相關的資源，以免未使用的資源產生費用。此步驟中的資源名稱是本教學課程中建議的名稱 (例如，**`ecs-demo-codedeploy-app`**您的 CodeDeploy 應用程式名稱)。如果您使用不同的名稱，請務必在清理時使用這些名稱。

清除教學課程資源

1. 使用delete-deployment-group命令刪除部 CodeDeploy 署群組。

```
aws deploy delete-deployment-group --application-name ecs-demo-deployment-group --
deployment-group-name ecs-demo-dg --region aws-region-id
```

2. 使用刪除應用程式命令刪除應用程式。 CodeDeploy

```
aws deploy delete-application --application-name ecs-demo-deployment-group --
region aws-region-id
```

3. 使用刪除函數命令刪除 Lambda 掛鉤函數。

```
aws lambda delete-function --function-name AfterAllowTestTraffic
```

4. 使用命delete-log-group令刪除您的 CloudWatch 日誌群組。

```
aws logs delete-log-group --log-group-name /aws/lambda/AfterAllowTestTraffic
```

教學課程：使用 CodeDeploy 和 AWS 無伺服器應用程式模型部署更新的 Lambda 函數

AWS SAM 是建置無伺服器應用程式的開放原始碼架構。它將 AWS SAM 範本中的 YAML 語法轉換並擴充為 AWS CloudFormation 語法，以建置無伺服器應用程式，例如 Lambda 函數。如需詳細資訊，請參閱什麼是 AWS 無伺服器應用程式模型？

在本教學課程中，您會使用 AWS SAM 建立可執行下列作業的解決方案：

- 建立您的 Lambda 函數。
- 建立您的 CodeDeploy 應用程式和部署群組。
- 建立兩個 Lambda 函數，在 CodeDeploy生命週期掛接期間執行部署驗證測試
- 偵測 Lambda 函數何時更新。Lambda 函數的更新會觸發部署 CodeDeploy ，會逐步將生產流量從 Lambda 函數的原始版本轉移到更新版本。

> **Note**
>
> 本教學要求您建立資源，可能會對您的 AWS 帳戶收費。這些措施包括可能的費用 CodeDeploy CloudWatch, Amazon, 和 AWS Lambda. 如需詳細資訊，請參閱CodeDeploy 定價、Amazon CloudWatch 定價和AWS Lambda 定價。

主題

必要條件

若要完成此教學課程，您必須先：

- 完成「開始使用 CodeDeploy」中的步驟。
- 安裝 AWS Serverless Application Model CLI。如需相關資訊，請參閱安裝 AWS SAM CLI。
- 建立 S3 儲存貯體。 AWS SAM 會將 S AWS AM 範本中參照的成品上傳至此值區。

步驟 1：設定基礎架構

本主題說明如何使用 AWS SAM 為 AWS SAM 範本和 Lambda 函數建立檔案。然後，您可以使用 AWS SAM `package`和`deploy`命令在基礎結構中產生元件。當您的基礎設施準備就緒時，您會擁有 CodeDeploy 應用程式和部署群組、要更新和部署的 Lambda 函數，以及兩個 Lambda 函數，其中包含在部署 Lambda 函數時執行的驗證測試。完成後，您可以使 AWS CloudFormation 用在 Lambda 主控台中檢視元件，或使用 AWS CLI 來測試 Lambda 函數。

主題

- 建立您的檔案
- Package S AWS AM 應用程式
- 部署 S AWS AM 應用程式
- (選擇性) 檢查及測試您的基礎架構

建立您的檔案

若要建立您的基礎設施，您必須建立以下檔案：

- `template.yml`
- `myDateTimeFunction.js`
- `beforeAllowTraffic.js`
- `afterAllowTraffic.js`

主題

- 建立您的 AWS SAM 範本
- 為您的 Lambda 函數建立檔案
- 為您的 BeforeAllowTraffic Lambda 函數建立檔案

- 為您的 AfterAllowTraffic Lambda 函數建立檔案

建立您的 AWS SAM 範本

建立指定基礎結構中元件的 AWS SAM 範本檔案。

若要建立您的 AWS SAM 範本

1. 建立名為 SAM-Tutorial 的目錄。
2. 在 SAM-Tutorial 目錄中，建立名為 template.yml 的檔案。
3. 將下列 YAML 程式碼複製到 template.yml 中。這是您的 AWS SAM 範本。

```
AWSTemplateFormatVersion : '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: A sample SAM template for deploying Lambda functions.

Resources:
# Details about the myDateTimeFunction Lambda function
  myDateTimeFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: myDateTimeFunction.handler
      Runtime: nodejs18.x
# Instructs your myDateTimeFunction is published to an alias named "live".
      AutoPublishAlias: live
# Grants this function permission to call lambda:InvokeFunction
      Policies:
        - Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
              - "lambda:InvokeFunction"
            Resource: '*'
      DeploymentPreference:
# Specifies the deployment configuration
          Type: Linear10PercentEvery1Minute
# Specifies Lambda functions for deployment lifecycle hooks
          Hooks:
            PreTraffic: !Ref beforeAllowTraffic
            PostTraffic: !Ref afterAllowTraffic

# Specifies the BeforeAllowTraffic lifecycle hook Lambda function
```

```
  beforeAllowTraffic:
    Type: AWS::Serverless::Function
    Properties:
      Handler: beforeAllowTraffic.handler
      Policies:
        - Version: "2012-10-17"
# Grants this function permission to call
 codedeploy:PutLifecycleEventHookExecutionStatus
          Statement:
          - Effect: "Allow"
            Action:
              - "codedeploy:PutLifecycleEventHookExecutionStatus"
            Resource:
              !Sub 'arn:aws:codedeploy:${AWS::Region}:
${AWS::AccountId}:deploymentgroup:${ServerlessDeploymentApplication}/*'
        - Version: "2012-10-17"
# Grants this function permission to call lambda:InvokeFunction
          Statement:
          - Effect: "Allow"
            Action:
              - "lambda:InvokeFunction"
            Resource: !Ref myDateTimeFunction.Version
      Runtime: nodejs18.x
# Specifies the name of the Lambda hook function
      FunctionName: 'CodeDeployHook_beforeAllowTraffic'
      DeploymentPreference:
        Enabled: false
      Timeout: 5
      Environment:
        Variables:
          NewVersion: !Ref myDateTimeFunction.Version

# Specifies the AfterAllowTraffic lifecycle hook Lambda function
  afterAllowTraffic:
    Type: AWS::Serverless::Function
    Properties:
      Handler: afterAllowTraffic.handler
      Policies:
        - Version: "2012-10-17"
          Statement:
# Grants this function permission to call
 codedeploy:PutLifecycleEventHookExecutionStatus
          - Effect: "Allow"
            Action:
```

```
              - "codedeploy:PutLifecycleEventHookExecutionStatus"
            Resource:
              !Sub 'arn:aws:codedeploy:${AWS::Region}:
${AWS::AccountId}:deploymentgroup:${ServerlessDeploymentApplication}/*'
        - Version: "2012-10-17"
          Statement:
# Grants this function permission to call lambda:InvokeFunction
          - Effect: "Allow"
            Action:
              - "lambda:InvokeFunction"
            Resource: !Ref myDateTimeFunction.Version
      Runtime: nodejs18.x
# Specifies the name of the Lambda hook function
      FunctionName: 'CodeDeployHook_afterAllowTraffic'
      DeploymentPreference:
        Enabled: false
      Timeout: 5
      Environment:
        Variables:
          NewVersion: !Ref myDateTimeFunction.Version
```

此範本指定下列項目。如需詳細資訊，請參閱AWS SAM 範本概念。

一個名為 Lambda 函數 **myDateTimeFunction**

發佈此 Lambda 函數時，範本中的AutoPublishAlias行會將其連結至名為的別名live。在本教學課程稍後，此函數的更新會透 AWS CodeDeploy 過逐步將生產流量從原始版本轉移到更新版本來觸發部署。

兩個 Lambda 部署驗證功能

下列 Lambda 函數會在 CodeDeploy 生命週期掛接期間執行。函數中的程式碼會驗證已更新的 myDateTimeFunction 是否部署完成。驗證測試的結果會傳遞給 CodeDeploy 使用其 PutLifecycleEventHookExecutionStatus API 方法。如果驗證測試失敗，部署會失敗並轉返。

- CodeDeployHook_beforeAllowTraffic 會在 BeforeAllowTraffic 勾點期間執行。
- CodeDeployHook_afterAllowTraffic 會在 AfterAllowTraffic 勾點期間執行。

兩個函數的名稱以 CodeDeployHook_ 開頭。此CodeDeployRoleForLambda角色只允許在名稱以此前置詞開頭的 Lambda 函數中呼叫 Lambda invoke 方法。如需詳細資訊，請參閱

CodeDeploy API 參考PutLifecycleEventHookExecutionStatus中的AppSpec AWS Lambda 部署的「掛鉤」部分和。

自動偵測更新的 Lambda 函數

`AutoPublishAlias` 一詞告知框架偵測 `myDateTimeFunction` 函數何時變更，然後使用 `live` 別名部署此函數。

部署組態

部署組態會決定 CodeDeploy應用程式將流量從原始版本的 Lambda 函數轉移到新版本的速率。此範本指定預先定義的部署組態 `Linear10PercentEvery1Minute`。

> **Note**
>
> 您無法在 AWS SAM 範本中指定自訂部署組態。如需詳細資訊，請參閱 Create a Deployment Configuration。

部署生命週期勾點函數

`Hooks` 區段指定在生命週期事件勾點期間執行的函數。`PreTraffic` 指定在 `BeforeAllowTraffic` 勾點期間執行的函數。`PostTraffic` 指定在 `AfterAllowTraffic` 勾點期間執行的函數。

讓 Lambda 叫用另一個 Lambda 函數的許可

指定的`lambda:InvokeFunction`權限會授與 S AWS AM 應用程式所使用的角色叫用 Lambda 函數的權限。
當`CodeDeployHook_beforeAllowTraffic`和`CodeDeployHook_afterAllowTraffic`函數在驗證測試期間叫用部署的 Lambda 函數時，這是必需的。

為您的 Lambda 函數建立檔案

為您稍後在此教學課程中更新和部署的函數建立檔案。

> **Note**
>
> Lambda 函數可以使用 AWS Lambda. 如需詳細資訊，請參閱 AWS Lambda 執行時間。

若要建立您的 Lambda 函數

1. 建立文字檔案，並在 SAM-Tutorial 目錄中儲存為 myDateTimeFunction.js。
2. 將下列 Node.js 程式碼複製到 myDateTimeFunction.js 中。

```
'use strict';

    exports.handler = function(event, context, callback) {

      if (event.body) {
        event = JSON.parse(event.body);
      }

      var sc; // Status code
      var result = ""; // Response payload

      switch(event.option) {
        case "date":
          switch(event.period) {
            case "yesterday":
              result = setDateResult("yesterday");
              sc = 200;
              break;
            case "today":
              result = setDateResult();
              sc = 200;
              break;
            case "tomorrow":
              result = setDateResult("tomorrow");
              sc = 200;
              break;
            default:
              result = {
                "error": "Must specify 'yesterday', 'today', or 'tomorrow'."
              };
              sc = 400;
              break;
          }
          break;

    /*      Later in this tutorial, you update this function by uncommenting
            this section. The framework created by AWS SAM detects the update
```

```
            and triggers a deployment by CodeDeploy. The deployment shifts
            production traffic to the updated version of this function.

            case "time":
            var d = new Date();
            var h = d.getHours();
            var mi = d.getMinutes();
            var s = d.getSeconds();

            result = {
              "hour": h,
              "minute": mi,
              "second": s
            };
            sc = 200;
            break;
    */
          default:
            result = {
              "error": "Must specify 'date' or 'time'."
            };
            sc = 400;
          break;
      }

      const response = {
        statusCode: sc,
        headers: { "Content-type": "application/json" },
        body: JSON.stringify( result )
      };

      callback(null, response);

      function setDateResult(option) {

        var d = new Date(); // Today
        var mo; // Month
        var da; // Day
        var y; // Year

        switch(option) {
          case "yesterday":
            d.setDate(d.getDate() - 1);
            break;
```

```
          case "tomorrow":
            d.setDate(d.getDate() + 1);
          default:
           break;
        }

        mo = d.getMonth() + 1; // Months are zero offset (0-11)
        da = d.getDate();
        y = d.getFullYear();

        result = {
          "month": mo,
          "day": da,
          "year": y
        };

        return result;
      }
    };
```

Lambda 函數會傳回昨天、今天或明天的日、月和年。稍後在本教學課程中，您會取消註解程式碼，以更新函數來傳回您所指定的日期或時間的相關資訊 (例如，年、月、日，或目前的時、分、秒)。通過 AWS SAM 檢測和部署功能的更新版本創建的框架。

> **Note**
>
> 此 Lambda 函數也用於 AWS Cloud9 教學課程中。 AWS Cloud9 是一個基於雲的集成開發環境。如需有關如何在中建立、執行、更新和偵錯此函式的詳細資訊 AWS Cloud9，請參閱〈AWS Lambda 〉的教學課程 AWS Cloud9。

為您的 BeforeAllowTraffic Lambda 函數建立檔案

為您的`beforeAllowTraffic`鉤子 Lambda 函數創建文件。

1. 建立文字檔案，並在 `SAM-Tutorial` 目錄中儲存為 `beforeAllowTraffic.js`。
2. 將下列 Node.js 程式碼複製到 `beforeAllowTraffic.js` 中。此函數會在您的部署 `BeforeAllowTraffic` 勾點期間執行。

```
'use strict';
```

```
   const AWS = require('aws-sdk');
   const codedeploy = new AWS.CodeDeploy({apiVersion: '2014-10-06'});
   var lambda = new AWS.Lambda();

   exports.handler = (event, context, callback) => {

    console.log("Entering PreTraffic Hook!");

    // Read the DeploymentId and LifecycleEventHookExecutionId from the event
payload
     var deploymentId = event.DeploymentId;
    var lifecycleEventHookExecutionId = event.LifecycleEventHookExecutionId;

    var functionToTest = process.env.NewVersion;
    console.log("BeforeAllowTraffic hook tests started");
    console.log("Testing new function version: " + functionToTest);

    // Create parameters to pass to the updated Lambda function that
    // include the newly added "time" option. If the function did not
    // update, then the "time" option is invalid and function returns
    // a statusCode of 400 indicating it failed.
    var lambdaParams = {
     FunctionName: functionToTest,
     Payload: "{\"option\": \"time\"}",
     InvocationType: "RequestResponse"
    };

    var lambdaResult = "Failed";
    // Invoke the updated Lambda function.
    lambda.invoke(lambdaParams, function(err, data) {
     if (err){ // an error occurred
      console.log(err, err.stack);
      lambdaResult = "Failed";
     }
     else{ // successful response
      var result = JSON.parse(data.Payload);
      console.log("Result: " +  JSON.stringify(result));
         console.log("statusCode: " + result.statusCode);

         // Check if the status code returned by the updated
         // function is 400. If it is, then it failed. If
         // is not, then it succeeded.
      if (result.statusCode != "400"){
```

```
            console.log("Validation succeeded");
        lambdaResult = "Succeeded";
          }
          else {
            console.log("Validation failed");
          }

       // Complete the PreTraffic Hook by sending CodeDeploy the validation status
       var params = {
        deploymentId: deploymentId,
        lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
        status: lambdaResult // status can be 'Succeeded' or 'Failed'
       };

       // Pass CodeDeploy the prepared validation test results.
       codedeploy.putLifecycleEventHookExecutionStatus(params, function(err, data)
{
        if (err) {
         // Validation failed.
         console.log("CodeDeploy Status update failed");
         console.log(err, err.stack);
         callback("CodeDeploy Status update failed");
        } else {
         // Validation succeeded.
         console.log("CodeDeploy status updated successfully");
         callback(null, "CodeDeploy status updated successfully");
        }
       });
      }
     });
    }
```

為您的 AfterAllowTraffic Lambda 函數建立檔案

為您的`afterAllowTraffic`鉤子 Lambda 函數創建文件。

1. 建立文字檔案，並在 SAM-Tutorial 目錄中儲存為 `afterAllowTraffic.js`。
2. 將下列 Node.js 程式碼複製到 `afterAllowTraffic.js` 中。此函數會在您的部署 `AfterAllowTraffic` 勾點期間執行。

```
'use strict';
```

```
    const AWS = require('aws-sdk');
    const codedeploy = new AWS.CodeDeploy({apiVersion: '2014-10-06'});
    var lambda = new AWS.Lambda();

    exports.handler = (event, context, callback) => {

     console.log("Entering PostTraffic Hook!");

     // Read the DeploymentId and LifecycleEventHookExecutionId from the event
payload
      var deploymentId = event.DeploymentId;
     var lifecycleEventHookExecutionId = event.LifecycleEventHookExecutionId;

     var functionToTest = process.env.NewVersion;
     console.log("AfterAllowTraffic hook tests started");
     console.log("Testing new function version: " + functionToTest);

     // Create parameters to pass to the updated Lambda function that
     // include the original "date" parameter. If the function did not
     // update as expected, then the "date" option might be invalid. If
     // the parameter is invalid, the function returns
     // a statusCode of 400 indicating it failed.
     var lambdaParams = {
      FunctionName: functionToTest,
      Payload: "{\"option\": \"date\", \"period\": \"today\"}",
      InvocationType: "RequestResponse"
     };

     var lambdaResult = "Failed";
     // Invoke the updated Lambda function.
     lambda.invoke(lambdaParams, function(err, data) {
      if (err){ // an error occurred
       console.log(err, err.stack);
       lambdaResult = "Failed";
      }
      else{ // successful response
       var result = JSON.parse(data.Payload);
       console.log("Result: " +  JSON.stringify(result));
          console.log("statusCode: " + result.statusCode);

          // Check if the status code returned by the updated
          // function is 400. If it is, then it failed. If
          // is not, then it succeeded.
       if (result.statusCode != "400"){
```

```
            console.log("Validation of time parameter succeeded");
        lambdaResult = "Succeeded";
          }
          else {
            console.log("Validation failed");
          }

       // Complete the PostTraffic Hook by sending CodeDeploy the validation status
       var params = {
        deploymentId: deploymentId,
        lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
        status: lambdaResult // status can be 'Succeeded' or 'Failed'
       };

       // Pass CodeDeploy the prepared validation test results.
       codedeploy.putLifecycleEventHookExecutionStatus(params, function(err, data)
{
        if (err) {
         // Validation failed.
         console.log("CodeDeploy Status update failed");
         console.log(err, err.stack);
         callback("CodeDeploy Status update failed");
        } else {
         // Validation succeeded.
         console.log("CodeDeploy status updated successfully");
         callback(null, "CodeDeploy status updated successfully");
        }
       });
      }
     });
    }
```

Package S AWS AM 應用程式

您的 SAM-Tutorial 目錄現在應該有四個檔案：

- beforeAllowTraffic.js
- afterAllowTraffic.js
- myDateTimeFunction.js
- template.yml

您現在可以使用 AWS SAM 命sam package令為 Lambda 函數和 CodeDeploy 應用程式建立和封裝成品。成品會上傳至 S3 儲存貯體。命令的輸出是名為 `package.yml` 的新檔案。在下一個步驟中，S AWS AM sam deploy 命令會使用此檔案。

> **Note**
>
> 如需有關sam package命令的詳細資訊，請參閱AWS Serverless Application Model 開發人員指南中的 AWS SAM CLI 命令參考。

在 `SAM-Tutorial` 目錄中執行下列命令。

```
sam package \
  --template-file template.yml \
  --output-template-file package.yml \
  --s3-bucket your-S3-bucket
```

對於`s3-bucket`參數，請指定您建立的 Amazon S3 儲存貯體做為本教學的先決條件。`output-template-file`指定 S AWS AM sam deploy 命令所使用的新檔案的名稱。

部署 S AWS AM 應用程式

使用 S AWS AM sam deploy 命令搭配`package.yml`檔案建立您的 Lambda 函數，以及使用的 CodeDeploy 應用程式和部署群組 AWS CloudFormation。

> **Note**
>
> 如需有關sam deploy命令的詳細資訊，請參閱AWS Serverless Application Model 開發人員指南中的 AWS SAM CLI 命令參考。

在 `SAM-Tutorial` 目錄中執行下列命令。

```
sam deploy \
  --template-file package.yml \
  --stack-name my-date-time-app \
  --capabilities CAPABILITY_IAM
```

需要`--capabilities CAPABILITY_IAM`參數才能授權 AWS CloudFormation 才能建立 IAM 角色。

(選擇性) 檢查及測試您的基礎架構

本主題說明如何檢視基礎設施元件及測試 Lambda 函數。

在執行 **sam deploy** 後查看堆疊的結果

1. 請在以下位置開啟 AWS CloudFormation 主控台。 https://console.aws.amazon.com/cloudformation
2. 在導覽窗格中，選擇 Stacks (堆疊)。my-date-time-app 堆疊會顯示在頂端。
3. 選擇 Events (事件) 標籤，以查看已完成的事件。您可以在堆疊建立進行時檢視事件。堆疊建立完成時，您可以查看所有堆疊建立事件。
4. 選取堆疊後，選擇 Resources (資源)。在「類型」欄中，您可以看到您的 Lambda 函數myDateTimeFunctionCodeDeployHook_beforeAllowTraffic、、和CodeDeployHook_afterAllowTraffic。每個 Lambda 函數的實體 ID 資料行都包含一個連結，可在 Lambda 主控台中檢視函數。

   > **Note**
   >
   > myDateTimeFunctionLambda 函數的名稱前面加上 AWS CloudFormation 堆棧的名稱，並在其中添加了一個標識符，因此看起來像。my-date-time-app-myDateTimeFunction-123456ABCDEF

5. 請在以下位置開啟 CodeDeploy 主控台。 https://console.aws.amazon.com/codedeploy/
6. 在導覽窗格中，展開 Deploy (部署)，然後選擇 Applications (應用程式)。
7. 您應該會看到以開頭的名稱建立 AWS CloudFormation 的新應 CodeDeploy 用程式my-date-time-app-ServerlessDeploymentApplication。選擇此應用程式。
8. 您應該會看到名稱開頭為 my-date-time-app-myDateTimeFunctionDeploymentGroup 的部署群組。選擇此部署群組。

   在「部署組態」下，您應該會看到CodeDeployDefault。 LambdaLinear十PercentEvery一分鐘。

（可選）來測試你的功能（控制台）

1. 請在以下位置開啟 AWS Lambda 主控台。 https://console.aws.amazon.com/lambda/
2. 從導覽窗格中，選擇您的 my-date-time-app-myDateTimeFunction 函數。在主控台中，其名稱包含識別符，因此看起來像 my-date-time-app-myDateTimeFunction-123456ABCDEF。

3. 選擇 測試。
4. 在 Event name (事件名稱) 中，輸入測試事件的名稱。
5. 為您的測試事件輸入下列內容，然後選擇 Create (建立)。

```
{
  "option": "date",
  "period": "today"
}
```

6. 選擇 測試。您在測試事件清單中應該只會看到您的測試事件。

   對於 Execution result (執行結果)，您應該會看到 succeeded (成功)。
7. 在 Execution result (執行結果) 下，展開 Details (詳細資訊) 以查看結果。您應該會看到目前的年、月、日。

（可選）來測試你的功能（AWS CLI）

1. 找到 Lambda 函數的 ARN。當您檢視函數時，它會顯示在 Lambda 主控台的頂端。
2. 執行下列命令。*your-function-arn*使用函數 ARN 取代。

```
aws lambda invoke \
--function your-function-arn \
--cli-binary-format raw-in-base64-out \
--payload "{\"option\": \"date\", \"period\": \"today\"}" out.txt
```

3. 開啟 `out.txt` 以確認結果包含目前的年、月、日。

步驟 2：更新 Lambda 函數

在本主題中，您會更新 `myDateTimeFunction.js` 檔案。在下一個步驟中，您將使用此檔案部署更新的函數。這會透過 CodeDeploy 將生產流量從目前版本的 Lambda 函數轉移到更新版本來觸發部署。

若要更新您的 Lambda 函數

1. 打開 `myDateTimeFunction.js`.
2. 移除兩個註解標記（「`/*`」和「`*/`」），以及 `switch` 區塊中名為 `time` 的 `case` 開頭和結尾的說明文字。

取消註解的程式碼可讓您將新的參數 time 傳遞至函數。如果您將 time 傳遞給更新的函數，它會傳回目前的 hour、minute 和 second。

3. 儲存 myDateTimeFunction.js。它看起來應該如下所示：

```
'use strict';

exports.handler = function(event, context, callback) {

  if (event.body) {
    event = JSON.parse(event.body);
  }

  var sc; // Status code
  var result = ""; // Response payload

  switch(event.option) {
    case "date":
      switch(event.period) {
        case "yesterday":
          result = setDateResult("yesterday");
          sc = 200;
          break;
        case "today":
          result = setDateResult();
          sc = 200;
          break;
        case "tomorrow":
          result = setDateResult("tomorrow");
          sc = 200;
          break;
        default:
          result = {
            "error": "Must specify 'yesterday', 'today', or 'tomorrow'."
          };
          sc = 400;
          break;
      }
      break;
      case "time":
        var d = new Date();
        var h = d.getHours();
        var mi = d.getMinutes();
```

```
        var s = d.getSeconds();

        result = {
          "hour": h,
          "minute": mi,
          "second": s
        };
        sc = 200;
        break;

      default:
        result = {
          "error": "Must specify 'date' or 'time'."
        };
        sc = 400;
      break;
  }

  const response = {
    statusCode: sc,
    headers: { "Content-type": "application/json" },
    body: JSON.stringify( result )
  };

  callback(null, response);

  function setDateResult(option) {

    var d = new Date(); // Today
    var mo; // Month
    var da; // Day
    var y; // Year

    switch(option) {
      case "yesterday":
        d.setDate(d.getDate() - 1);
        break;
      case "tomorrow":
        d.setDate(d.getDate() + 1);
      default:
       break;
    }

    mo = d.getMonth() + 1; // Months are zero offset (0-11)
```

```
    da = d.getDate();
    y = d.getFullYear();

    result = {
      "month": mo,
      "day": da,
      "year": y
    };

    return result;
  }
};
```

步驟 3：部署更新的 Lambda 函數

在此步驟中，您可以使用更新程式myDateTimeFunction.js來更新和啟動 Lambda 函數的部署。您可以在 CodeDeploy 或 AWS Lambda 主控台中監視部署進度。

AWS SAM 範本中的這一AutoPublishAlias: live行會讓您的基礎結構偵測使用live別名之函數的更新。函數的更新會觸發部署 CodeDeploy ，將生產流量從原始版本的函數轉移到更新版本。

sam package和命sam deploy令可用來更新和觸發 Lambda 函數的部署。您在 Package S AWS AM 應用程式 和部署 S AWS AM 應用程式中執行過這些命令。

若要部署您更新的 Lambda 函數

1. 在 SAM-Tutorial 目錄中執行下列命令。

```
sam package \
  --template-file template.yml \
  --output-template-file package.yml  \
  --s3-bucket your-S3-bucket
```

   這會建立一組新的成品，參考 S3 儲存貯體中更新的 Lambda 函數。

2. 在 SAM-Tutorial 目錄中執行下列命令。

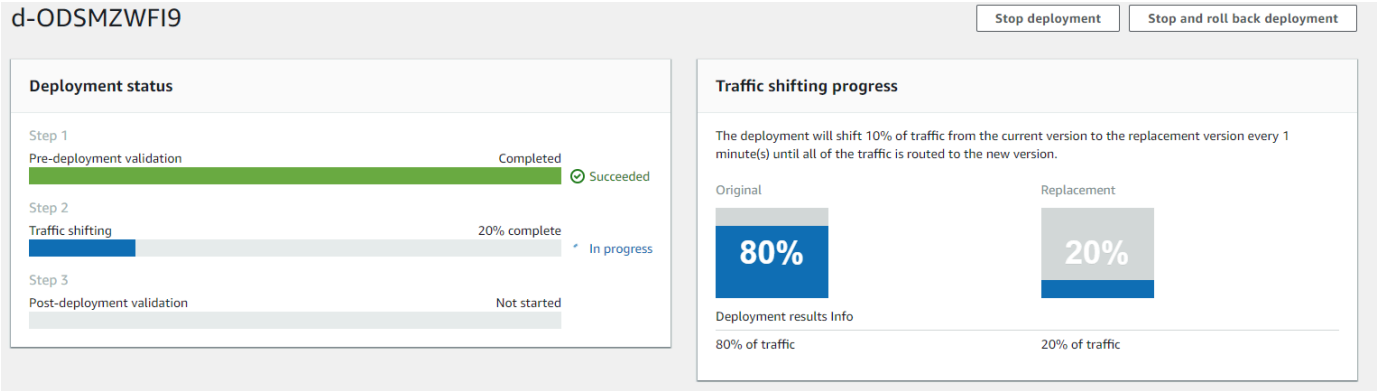
```
sam deploy \
  --template-file package.yml \
  --stack-name my-date-time-app \
  --capabilities CAPABILITY_IAM
```

因為堆疊名稱仍然是`my-date-time-app`，所以請 AWS CloudFormation 認識到這是堆疊更新。若要檢視更新的堆疊，請返回 AWS CloudFormation 主控台，然後從導覽窗格中選擇 [堆疊]。

(選擇性) 以檢視部署期間的流量 (CodeDeploy 主控台)

1. 請在以下位置開啟 CodeDeploy 主控台。 https://console.aws.amazon.com/codedeploy/
2. 在瀏覽窗格中，展開 [應用程式]，然後選擇您的 my-date-time-app- 應用 ServerlessDeploymentApplication程式。
3. 在 Deployment groups (部署群組) 中，選擇您應用程式的部署群組。其狀態應為 In progress (進行中)。
4. 在 Deployment group history (部署群組歷史記錄) 中，選擇進行中的部署。

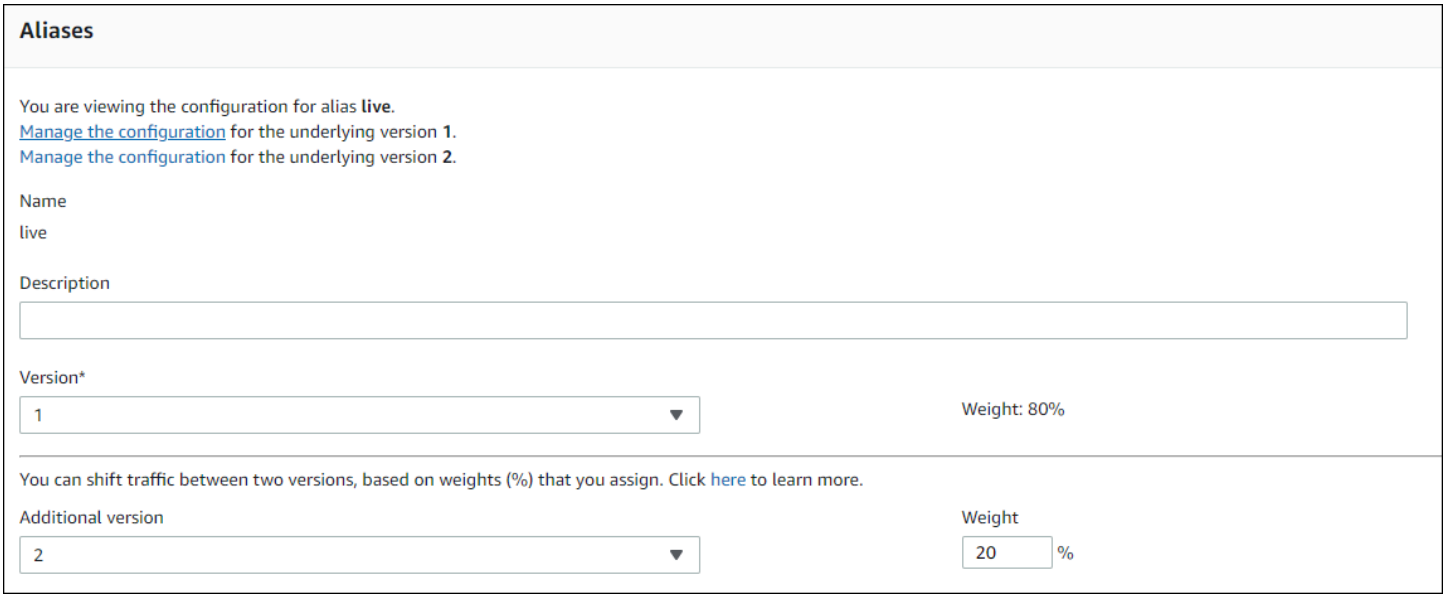
   此頁面上的 Traffic shifting (流量轉移) 進度列及 Original (原始) 和 Replacement (取代) 方塊中的百分比會顯示其進度。



(選用) 以檢視部署期間的流量 (Lambda 主控台)

1. 請在以下位置開啟 AWS Lambda 主控台。 https://console.aws.amazon.com/lambda/
2. 從導覽窗格中，選擇您的 `my-date-time-app-myDateTimeFunction` 函數。在主控台中，其名稱包含識別符，因此看起來像 `my-date-time-app-myDateTimeFunction-123456ABCDEF`。
3. 選擇 [別名]，然後選擇 [即時]。

原始函數版本 (版本 1) 和已更新的函數版本 (版本 2) 旁的權重會顯示載入此 AWS Lambda 主控台頁面時，每個版本接到多少流量。頁面不會隨著時間更新權重。如果您每分鐘重新整理一次頁面，版本 1 的權重會減少 10%，而版本 2 的權重會增加 10%，直到版本 2 的權重達到 100 為止。



步驟 4：檢視部署結果

在此步驟中，您將檢視部署的結果。如果部署成功，您可以確認更新的 Lambda 函數是否接收生產流量。如果部署失敗，您可以使用 CloudWatch Logs 來檢視 Lambda 函數中的驗證測試輸出，該函數會在部署的生命週期期間執行。

主題

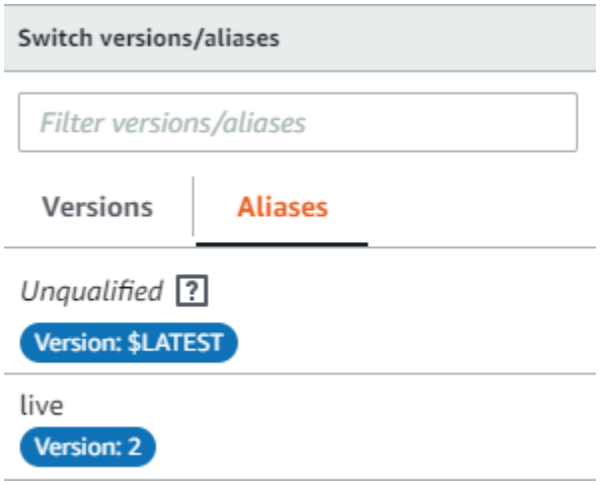
- 測試您部署的功能
- 檢視 CloudWatch 記錄檔中的勾點事件

測試您部署的功能

此命sam deploy令會更新 L `my-date-time-app-myDateTimeFunction` ambda 函數。函數版本已更新為 2 並新增至 `live` 別名。

若要在 Lambda 主控台中查看更新

1. 請在以下位置開啟 AWS Lambda 主控台。 https://console.aws.amazon.com/lambda/
2. 從導覽窗格中，選擇 `my-date-time-app-myDateTimeFunction` 函數。在主控台中，其名稱包含識別符，因此看起來像 `my-date-time-app-myDateTimeFunction-123456ABCDEF`。
3. 選擇 Qualifiers (限定詞)，然後選擇 Aliases (別名)。部署完成後 (約 10 分鐘)，針對 `live` 別名，您應該會看到 Version: 2 (版本：2)。



4. 在 Function code (函數原始程式碼) 中，檢視函數的來源碼。您的變更應該會出現。
5. (選用) 您可以使用步驟 2：更新 Lambda 函數中的測試說明來測試已更新的函數。使用下列承載建立新的測試事件，然後確認結果包含目前的時、分、秒。

```
{
    "option": "time"
  }
```

若要使用 AWS CLI 來測試您更新的函數，請執行下列命令，然後開啟`out.txt`以確認結果包含目前的小時、分鐘和秒。

```
aws lambda invoke --function your-function-arn --payload "{\"option\": \"time\"}"
 out.txt
```

> **Note**
>
> 如果您在部署完成之前使用 AWS CLI 來測試函數，可能會收到非預期的結果。這是因為每分鐘會 CodeDeploy 逐步將 10% 的流量轉移到更新版本。在部署期間，有些流量仍指向原始版本，因此 `aws lambda invoke` 可能會使用原始版本。10 分鐘後，部署完成，所有流量都指向新版本的函數。

檢視 CloudWatch 記錄檔中的勾點事件

在`BeforeAllowTraffic`掛接期間， CodeDeploy 執行您的 `CodeDeployHook_beforeAllowTraffic` Lambda 函數。在`AfterAllowTraffic`掛接期間，

CodeDeploy 執行您的 `CodeDeployHook_afterAllowTraffic` Lambda 函數。每個函數都會執行驗證測試，以使用新的 `time` 參數呼叫更新版本的函數。如果 Lambda 函數更新成功，則該`time`選項不會導致錯誤，並且驗證成功。如果函數未更新，無法辨識的參數會導致錯誤且驗證失敗。這些驗證測試僅供示範之用。您撰寫自己的測試來驗證部署。您可以使用記 CloudWatch 錄主控台來檢視驗證測試。

若要檢視您的 CodeDeploy 勾點事件

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 從導覽窗格中，選擇 Logs (日誌)。
3. 從記錄群組清單中，選擇 /aws/羊肉/ _ 或 /aws/羊肉達/ CodeDeployHook _。beforeAllowTraffic CodeDeployHook afterAllowTraffic
4. 選擇日誌串流。您應該只會看到一個。
5. 展開事件以查看其詳細資訊。

步驟 5：清除

為了避免您在本教學課程中使用的資源進一步收費，請刪除 AWS SAM 範本建立的資源以及 Lambda 驗證函數建立的 CloudWatch 日誌。

若要刪除您的 AWS CloudFormation 堆疊

1. 請登入 AWS Management Console 並開啟 AWS CloudFormation 主控台，網址為 https://console.aws.amazon.com/cloudformation。
2. 在 Stacks (堆疊) 欄中，選擇您的 `my-date-time-app` 堆疊，然後選擇 Delete (刪除)。
3. 出現提示時，選擇 Delete stack (刪除堆疊)。Lambda 函數、 CodeDeploy應用程式和部署群組，以及由建立的 IAM 角色 AWS SAM 都會遭到刪除。

若要刪除記錄檔中的 CloudWatch 記錄

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 從導覽窗格中，選擇 Logs (日誌)。
3. 從記錄群組清單中，選擇 /aws/lambda/ CodeDeployHook _ 旁邊的按鈕。beforeAllowTraffic
4. 從 Actions (動作) 中，選擇 Delete log group (刪除日誌群組)，然後選擇 Yes, Delete (是，刪除)。
5. 從記錄群組清單中，選擇 /aws/lambda/ CodeDeployHook _ 旁邊的按鈕。afterAllowTraffic
6. 從 Actions (動作) 中，選擇 Delete log group (刪除日誌群組)，然後選擇 Yes, Delete (是，刪除)。

與 CodeDeploy 代理工作

AWS CodeDeploy 代理程式是一種軟體套件，只要在執行個體上安裝並設定該執行個體，就可以在 CodeDeploy 部署中使用該執行個體。

AWS 支援 CodeDeploy 代理程式的最新次要版本。目前最新的次要版本是 1.7.x。

> **Note**
>
> 僅當您部署到 EC2 /內部部署計算平台時，才需要 CodeDeploy 代理程式。使用 Amazon ECS 或 AWS Lambda 運算平台的部署不需要代理程式。

安裝代理程式時，組態檔案會置放於執行個體上。檔案會用於指定代理程式的運作方式。此組態檔案會指定目錄路徑和其他設定， AWS CodeDeploy 以便在與執行個體互動時使用。您可以在檔案中變更一部分的組態選項。如需有關使用 CodeDeploy 代理程式組態檔的資訊，請參閱CodeDeploy 用戶端組態參考。

如需有關使用 CodeDeploy 代理程式的詳細資訊，例如安裝、更新和驗證版本的步驟，請參閱管理 CodeDeploy 代理程式作。

主題

CodeDeploy 代理程式支援的作業系統

支援的 Amazon EC2 AMI 作業系統

CodeDeploy 代理程式已在下列 Amazon EC2 AMI 作業系統上進行測試：

- Amazon 系統 2023 (手臂, x86)
- Amazon Linux 2 (ARM、x86)
- Microsoft 視窗伺服器
- 紅帽企業版 Linux (RHEL) 9.x, 8.x, 7.x
- Ubuntu 服務器 22.04 LTS，20.04 LTS，18.04 LTS，16.04 LTS

該 CodeDeploy 代理程式可作為開放原始碼提供，以適應您的需求。它可以與其他 Amazon EC2 AMI 操作系統一起使用。如需詳細資訊，請移至中的CodeDeploy 代理程式儲存庫 GitHub。

支援的本機作業系統

CodeDeploy 代理程式已在下列內部部署作業系統上進行測試：

- Microsoft 視窗伺服器
- 紅帽企業版 Linux (RHEL) 9.x, 8.x, 7.x
- Ubuntu 服務器 22.04 LTS，20.04 LTS

該 CodeDeploy 代理程式可作為開放原始碼提供，以適應您的需求。它可搭配其他現場部署執行個體作業系統使用。如需詳細資訊，請移至中的CodeDeploy 代理程式儲存庫 GitHub。

CodeDeploy 代理程式的通訊協定和連接埠

CodeDeploy 代理程式透過連接埠 443 使用 HTTPS 進行輸出通訊。

當 CodeDeploy 代理程式在 EC2 執行個體上執行時，會使用 EC2 中繼資料端點擷取執行個體相關資訊。深入了解限制和授與執行個體中繼資料服務存取的相關資訊。

CodeDeploy代理程式的版本歷史記錄

您的執行個體必須執行受支援的 CodeDeploy 代理程式版本。目前支援的最低版本為 1.7.x。

> **Note**
>
> 我們建議使用最新版本的 CodeDeploy 代理程式。如果您遇到問題，請在聯絡 Sup AWS port 人員之前更新至最新版本。如需升級資訊，請參閱更新 CodeDeploy 代理程式。

下表列出 CodeDeploy 代理程式的所有版本，以及每個版本隨附的功能和增強功能。

版本	版本日期	詳細資訊
1.7.0	2024年3月6日	添加： CodeDeploy 代理:`disable_imds_v1:` 配置文件的配置設置。使用此設定可在發生 IMDSv2 錯誤時停用 IMDSv1 的後援功能。預設為 `false` (啟用後援)。如需詳細資訊，請參閱CodeDeploy 代理程式組態參考。 新增：Support RHEL 9 (RHEL 9) 作業系統。 補充：紅寶石版本 3.1 和 3.2 的 Ubuntu 服務器上的 Support。 修正：如果 CodeDeploy 代理配置文件加載失敗， CodeDeploy 代理現在會生成用戶友好的錯誤。 更改：在 Windows 的 CodeDeploy 代理程序中將紅寶石升級到 2.7.8-1。
1.6.0	2023 年 3 月 30 日	補充：Support 紅寶石 3.1，3.2。 新增：Support Amazon 系統 2023。 新增：Support 視窗伺服器 2022。 已變更：的預設設定現在`false`適用於 Windows 伺服器執行個體。`verbose`若要繼續在 Windows 上的記錄檔中列印偵錯訊息，您必須`verbose`將設定為`true`。 刪除：Support 視窗服務器 2016 年和視窗服務器 2012 R2。 已移除：Support Amazon Linux
1.5.0	2023 年 3 月 3 日	補充：Support 紅寶石 3. 新增：Support 修復：啟動後立即重新啟動 CodeDeploy 代理程式會導致代理程式懸置的問題。

版本	版本日期	詳細資訊
		已變更：如果 CodeDeploy 代理程式服務在執行勾點指令碼時意外重新啟動，代理程式現在會失敗主機部署。此修正程式可讓您避免在重試部署之前等待 70 分鐘逾時期間。 棄用通知： CodeDeploy 代理程式 1.5.0 是支援視窗伺服器 2016 年和視窗伺服器 2012 R2 的最新版本。 刪除：Support 在 Ubuntu 14.04 LTS CodeDeploy 代理，視窗服務器 2008 R2 和視窗服務器 2008 R2 32 位。
1.4.1	2022 年 12 月 6 日	修正：與日誌記錄相關的安全漏洞。 增強功能：改善輪詢主機命令時的記錄功能。

版本	版本日期	詳細資訊
1.4.0	2022 年 8 月 31 日	新增：Support 紅帽企業版 Linux 8. 新增：Support Windows CodeDeploy 代理程式上的長檔案路徑。若要啟用長檔案路徑，您需要設定適當的 Windows 登錄機碼，然後重新啟動代理程式。如需詳細資訊，請參閱 長文件路徑導致「沒有這樣的文件或目錄」錯誤。 修正：磁碟已滿時解壓縮作業的問題。 CodeDeploy 代理程式現在會偵測解壓縮的結束代碼 50，表示磁碟已滿，移除部分解壓縮的檔案，並引發例外狀況，將失敗張貼至 CodeDeploy伺服器。錯誤訊息會顯示為生命週期事件錯誤訊息，且主機層級部署將停止而不會卡住或逾時。 修正：會導致代理程式失敗的問題。 修正：在邊緣情況下，掛鉤會逾時的問題。沒有指令碼的勾點現在會繼續進行，不會再造成失敗或逾時。 已變更：已移除 CodeDeploy 代理程式`bin`目錄中的`update`指令碼，因為已不再使用。 已變更：視窗伺服器的 CodeDeploy 代理程式現在會捆綁 Ruby 2.7。 已變更：新增了新的環境變數，以供勾點指令碼使用，具體取決於部署服務包的來源 (Amazon S3 或 GitHub)。 如需詳細資訊，請參閱 掛接的環境變數可用性。 **Important** 棄用通知： CodeDeploy代理程式 1.4.0 是最後一個版本，其中包含 32 位元 Windows 伺服器的安裝程式。 棄用通知： CodeDeploy代理程式 1.4.0 是將支援視窗伺服器 2008 R2 的最後一個版本。

版本	版本日期	詳細資訊
		已移除：Support 下列 Amazon EC2 AMI 上的 CodeDeploy 代理程式：亞馬遜 Linux

版本	版本日期	詳細資訊
1.3.2	2021 年 5 月 6 日	**Important** CodeDeploy 代理程式 1.3.2 可解決 CVE-2018-1000201，這會影響執行代理程式的 Windows 主機。CVE 引用了紅寶石 FFI，這是代理程式的相依性。CodeDeploy 如果您的代理程式是透過 Amazon EC2 Systems Manager (SSM) 安裝的，且設定為自動更新，則不需要採取任何動作。否則，需要採取動作才能手動更新代理程式。如果要升級代理程式，請遵循在 Windows 伺服器上更新 CodeDeploy 代理程式中的指示。 已修正：在 Ubuntu 20.04 及更新版本上安裝 CodeDeploy 代理程式時出現的問題。 修復：由於相對路徑未正確處理，因此在提取壓縮文件時發生的間歇性問題。 新增：Support Windows 執行個體的 VPC 人雲端端點。AWS PrivateLink 補充： AppSpec 文件的改進，如下所述。 • 您現在可以在建立本端部署時指定 AppSpec 檔案的自訂檔案名稱。如需詳細資訊，請參閱 建立本機部署。 • 該 AppSpec 文件現在可以有一個`.yaml`文件擴展名。 • 您現在可以使用檔案中的新選用`file_exists_behavior` 設定覆寫已部署的 AppSpec檔案。如需詳細資訊，請參閱 AppSpec 「檔案」區段 (僅適用於 EC2 /內部部署)。 升級： CodeDeploy 現在使用 AWS SDK for Ruby 3.0。

版本	版本日期	詳細資訊
1.3.1	2020 年 12 月 22 日	修正：阻止內部部署執行個體啟動的 1.3.0 問題。
1.3.0	2020 年 11 月 10 日	**Important** 此版本已被棄用。 修正：刪除了不再使用的過期證書。 修正：從使用的代理卸載腳本中刪除了提示消息 AWS Systems Manager，以便更輕鬆地將主機或車隊降級到以前版本的代理程序。
1.2.1	2020 年 9 月 23 日	更改：從 v2 升級到 v3 的 AWS SDK for Ruby 依賴關係。 補充：Support IMDSv2。如果 IMDSv2 http 要求失敗，則包含對 IMDSv1 的無訊息後援。 已變更：更新了安全性修補程式的 Rake 和 Rubyzip 相依性。 修正：確保一個空的 PID 文件將返回一個狀態，No `CodeDeploy Agent Running` 並在代理啟動時清理 PID 文件。

版本	版本日期	詳細資訊
1.1.2	2020 年 8 月 4 日	補充：Support Ubuntu 服務器 19.10 和 20.04。 注意：版本 19.10 已到達其 end-of-life 日期，Ubuntu 或 CodeDeploy. 補充：Linux 和 Ubuntu 的內存效率改進，以更及時地釋放保留內存。 新增：與 Windows Server「靜音清理」的兼容性，這在某些情況下導致代理程序無響應。 新增：清理期間忽略非空目錄，以避免部署失敗。 新增：Support 洛杉磯 (LA) 的 AWS 本地區域。 新增：從執行個體中繼資料擷取 AZ，以提供 L AWS ocal Zones 的相容性。 補充：用戶現在可以在子目錄中提供存檔，不需要將其存儲在根目錄中。 新增：檢測到 Rubyzip 可能導致內存洩漏的問題。更新了 unzip 命令，以在使用 Rubyzip 之前先嘗試使用系統安裝的解壓縮公用程式。 新增：`:enable_auth_policy:` 作為代理配置設置。 已變更：解壓縮警告現在會被忽略，因此部署將繼續進行。

版本	版本日期	詳細資訊
1.1.0	2020 年 6 月 30 日	已更改： CodeDeploy代理的版本控制現在遵循 Ruby 標準版本控制約定。 已新增：安裝和更新命令的新參數，以允許從命令列安裝特定的代理程式版本。 已移除：已移除 Linux 和 Ubuntu 的 CodeDeploy 代理程式自動更新程式。如果要設定 CodeDeploy 代理程式的自動更新，請參閱使用安裝 CodeDeploy 代理程式 AWS Systems Manager。
1.0.1.1597	2018 年 11 月 15 日	增強： CodeDeploy 支持 增強： CodeDeploy 支持紅寶石 2.5。 增強功能： CodeDeploy 支援 FIPS 端點。如需 FIPS 端點的詳細資訊，請參閱 FIPS 140-2 概觀。如需可搭配使用的端點 CodeBuild，請參閱CodeDeploy區域和端點。
1.0.1.1518	2018 年 6 月 12 日	增強功能：修正當 CodeDeploy 代理程式在接受輪詢要求時關閉時造成錯誤的問題。 增強功能：已新增部署追蹤功能，可防止 CodeDeploy 代理程式在部署進行時關閉。 加強功能：改善刪除檔案時的效能。
1.0.1.1458	2018 年 3 月 6 日	注意：現已不支援此版本。如果您使用此版本，您的部署可能會失敗。 加強功能：改善憑證驗證，支援更多受信任的授權單位。 增強功能：修正在包含 BeforeInstall 生命週期事件的部署期間造成本機 CLI 失敗的問題。 增強功能：修正更新 CodeDeploy 代理程式時，可能導致作用中部署失敗的問題。

版本	版本日期	詳細資訊
1.0.1.1352	2017 年 11 月 16 日	注意：現已不支援此版本。如果您使用此版本，您的部署可能會失敗。 功能：推出了一項新功能，可在安裝 CodeDeploy 代理程式的本機電腦或執行個體上對 EC2/內部部署進行測試和偵錯。
1.0.1.1106	2017 年 5 月 16 日	注意：現已不支援此版本。如果您使用此版本，您的部署可能會失敗。 功能：在不屬於最近一次成功部署之應用程式修訂的目標位置中，針對處理內容的功能推出新支援。現有內容的部署選項現在包含保留內容、覆寫內容，或令部署失敗。 增強：使 CodeDeploy 代理與版本 2.9.2 的 AWS SDK for Ruby（aws-sdk-core2.9.2）兼容。
1.0.1.1095	2017 年 3 月 29 日	注意：現已不支援此版本。如果您使用此版本，您的部署可能會失敗。 增強：引入對中國（北京）地區 CodeDeploy 代理商的支持。 增強功能：當生命週期事件掛接叫用時，啟用 Puppet 在 Windows 伺服器執行個體上執行。 加強功能：改善 `untar` 操作的處理。
1.0.1.1067	2017 年 1 月 6 日	注意：現已不支援此版本。如果您使用此版本，您的部署可能會失敗。 加強功能：修訂許多錯誤訊息，包含造成部署失敗的更明確原因。 增強功能：已修正 CodeDeploy 代理程式無法識別要在部署期間部署的正確應用程式修訂版本的問題。 加強功能：還原在 `untar` 操作之前或之後使用 `pushd` 和 `popd` 的方法。

版本	版本日期	詳細資訊
1.0.1.1045	2016 年 11 月 21 日	注意：現已不支援此版本。如果您使用此版本，您的部署可能會失敗。 增強功能：使 CodeDeploy 代理與 AWS SDK for Ruby （aws-sdk-core2.6.11）的 2.6.11 版本兼容。
1.0.1.1037	2016 年 10 月 19 日	注意：現已不支援此版本。如果您使用此版本，您的部署可能會失敗。 Amazon Linux、RHEL 和 Ubuntu 伺服器執行個體的 CodeDeploy 代理程式已更新為下列變更。對於視窗伺服器執行個體，最新版本仍為 1.0.1.998。 加強功能：代理程式現在可以判斷執行個體上安裝的 Ruby 版本，並使用該版本呼叫 `codedeploy-agent` 指令碼。
1.0.1.1011.1	2016 年 8 月 17 日	注意：現已不支援此版本。如果您使用此版本，您的部署可能會失敗。 加強功能：因殼層支援問題，移除 1.0.1.1011 版推出的變更。此版本的代理程式在功能上與 2016 年 7 月 11 日發行的 1.0.1.998 版相同。
1.0.1.1011	2016 年 8 月 15 日	注意：現已不支援此版本。如果您使用此版本，您的部署可能會失敗。 Amazon Linux、RHEL 和 Ubuntu 伺服器執行個體的 CodeDeploy 代理程式已更新為下列變更。對於視窗伺服器執行個體，最新版本仍為 1.0.1.998。 功能：已新增在使用 systemd init 系統的作業系統上使用 bash 殼層叫用 CodeDeploy 代理程式的支援。 增強功能：已啟用 CodeDeploy 代理程式和代理程式更新程式中所有 Ruby 2.x 版本的支援。 CodeDeploy 更新的 CodeDeploy 代理程式不再僅依賴於 Ruby 2.0。(CodeDeploy 代理程式安裝程式的 deb 和 rpm 版本仍需要 Ruby 2.0)。

版本	版本日期	詳細資訊
1.0.1.998	2016 年 7 月 11 日	注意：現已不支援此版本。如果您使用此版本，您的部署可能會失敗。 增強功能：已修正使用 root 以外的使用者設定檔執行 CodeDeploy 代理程式的支援。名為 USER 的變數現已取代為 CODEDEPLOY_USER ，避免與環境變數產生衝突。
1.0.1.966	2016 年 6 月 16 日	注意：現已不支援此版本。如果您使用此版本，您的部署可能會失敗。 功能：支援使用 root 以外的使用者設定檔執行 CodeDeploy 代理程式。 增強功能：已修正支援指定您希望 CodeDeploy代理程式為部署群組封存的應用程式修訂數目。 增強：使 CodeDeploy 代理與 2.3 版本的 AWS SDK for Ruby（aws-sdk-core 2.3）兼容。 加強功能：修正部署期間的 UTF-8 編碼問題。 加強功能：改善識別程序名稱的準確度。
1.0.1.950	2016 年 3 月 24 日	注意：現已不支援此版本。如果您使用此版本，您的部署可能會失敗。 功能：新增安裝代理支援。 增強功能：已更新安裝指令碼，以便在已安裝最新版本時不下載 CodeDeploy 代理程式。
1.0.1.934	2016 年 2 月 11 日	注意：現已不支援此版本。如果您使用此版本，您的部署可能會失敗。 功能：引入支援指定 CodeDeploy代理程式為部署群組封存的應用程式修訂版本數目。

版本	版本日期	詳細資訊
1.0.1.880	2016 年 1 月 11 日	注意：現已不支援此版本，並可能導致部署失敗。 增強：使 CodeDeploy 代理與 AWS SDK for Ruby（aws-sdk-core 2.2）的 2.2 版本兼容。仍支援 2.1.2 版本。
1.0.1.854	2015 年 11 月 17 日	注意：現已不支援此版本。如果您使用此版本，您的部署可能會失敗。 功能：推出 SHA-256 雜湊演算法支援。 功能：推出 `.version` 檔案中的版本追蹤支援。 功能：讓部署群組 ID 可透過環境變數使用。 增強功能：已新增使用 Amazon 日誌監控 CodeDeploy 代理程式 CloudWatch 日誌的支援。

如需相關資訊，請參閱以下內容：

- 判斷代 CodeDeploy理程式的版本
- 安裝 CodeDeploy 代理程式

如需 CodeDeploy 代理程式版本的歷史記錄，請參閱上的發行存放庫 GitHub。

管理流 CodeDeploy 程

CodeDeploy 代理程式的所有 Linux 發行版本 (rpm 和 deb) 預設都使用 systemd 來管理代理程式程序。

不過，rpm 和 deb 發行版本都附有位於的啟動程序檔。`/etc/init.d/codedeploy-agent`視您使用的發行版本而定，當使用命令時`sudo service codedeploy-agent restart`，`/etc/init.d`可能會執行位於的指令碼以啟動代理程式處理程序，而不是`systemd`允許管理程序。在執行指令碼`/etc/init.d`是不可取的。

為了避免這個問題，對於支援的系統，`systemd`我們建議您將`systemctl`公用程式用於任何代理程式作業，而不要使用`service`命令。

例如，若要重新啟動 CodeDeploy 代理程式，請`sudo systemctl restart codedeploy-agent`使用公用`service`程式來取代對等的命令。

應用程式修訂和記錄檔清理

CodeDeploy 代理程式會在執行個體上封存修訂和記錄檔。 CodeDeploy 代理程式會清除這些人工因素以節省磁碟空間。

應用程式修訂版部署記錄檔：您可以使用代理程式組態檔中的:max_revision: 選項，透過輸入任何正整數來指定要封存的應用程式修訂版本數目。 CodeDeploy 也會封存這些修訂的記錄檔。所有其他的項目都會遭到刪除，除了最後一次成功部署的日誌檔案。該日誌檔案一律予以保留，即使失敗的部署數超過保留的修訂數也一樣。如果未指定任何值，則除了目前部署的修訂版本之外，還 CodeDeploy 會保留最新的五個修訂。

CodeDeploy 日誌：對於 Amazon Linux、Ubuntu 伺服器和 RHEL 執行個體， CodeDeploy 代理程式會輪替`/var/log/aws/codedeploy-agent`資料夾下的記錄檔。日誌檔案會在每天 00:00:00 (執行個體時間) 進行輪換。日誌檔案會在七天之後刪除。輪換日誌檔案的命名模式為 `codedeploy-agent.`*`YYYYMMDD`*`.log`。

CodeDeploy 代理程式安裝的檔案

CodeDeploy 代理程式會將修訂、部署歷程記錄和部署指令碼儲存在執行個體的根目錄中。此目錄的預設名稱和位置為：

`'/opt/codedeploy-agent/deployment-root'`適用於 Amazon Linux、Ubuntu 伺服器和 RHEL 執行個體。

`'C:\ProgramData\Amazon\CodeDeploy'`適用於視窗伺服器實例。

您可以使用 CodeDeploy 代理程式組態檔中的 root_dir 設定來設定目錄的名稱和位置。如需詳細資訊，請參閱 CodeDeploy 用戶端組態參考。

以下是根目錄下檔案及目錄結構的範例。結構假設有 N 個部署群組，且每個部署群組都包含 N 個部署。

```
|--deployment-root/
|-- deployment group 1 ID
|    |-- deployment 1 ID
|    |    |-- Contents and logs of the deployment's revision
|    |-- deployment 2 ID
|    |    |-- Contents and logs of the deployment's revision
```

```
|    |-- deployment N ID
|    |    |-- Contents and logs of the deployment's revision
|-- deployment group 2 ID
|    |-- deployment 1 ID
|    |    |-- bundle.tar
|    |    |-- deployment-archive
|    |    |    | -- contents of the deployment's revision
|    |    |-- logs
|    |    |    | -- scripts.log
|    |-- deployment 2 ID
|    |    |-- bundle.tar
|    |    |-- deployment-archive
|    |    |    | -- contents of the deployment's revision
|    |    |-- logs
|    |    |    | -- scripts.log
|    |-- deployment N ID
|    |    |-- bundle.tar
|    |    |-- deployment-archive
|    |    |    | -- contents of the deployment's revision
|    |    |-- logs
|    |    |    | -- scripts.log
|-- deployment group N ID
|    |-- deployment 1 ID
|    |    |-- Contents and logs of the deployment's revision
|    |-- deployment 2 ID
|    |    |-- Contents and logs of the deployment's revision
|    |-- deployment N ID
|    |    |-- Contents and logs of the deployment's revision
|-- deployment-instructions
|    |-- [deployment group 1 ID]_cleanup
|    |-- [deployment group 2 ID]_cleanup
|    |-- [deployment group N ID]_cleanup
|    |-- [deployment group 1 ID]_install.json
|    |-- [deployment group 2 ID]_install.json
|    |-- [deployment group N ID]_install.json
|    |-- [deployment group 1 ID]_last_successful_install
|    |-- [deployment group 2 ID]_last_successful_install
|    |-- [deployment group N ID]_last_successful_install
|    |-- [deployment group 1 ID]_most_recent_install
|    |-- [deployment group 2 ID]_most_recent_install
|    |-- [deployment group N ID]_most_recent_install
|-- deployment-logs
|    |-- codedeploy-agent-deployments.log
```

- 部署群組 ID 資料夾表示您的每個部署群組。部署群組目錄的名稱便是其 ID (例如，acde1916-9099-7caf-fd21-012345abcdef)。每個部署群組目錄都會為該部署群組中的每一次嘗試部署包含一個子目錄。

  您可以使用batch-get-deployments命令尋找部署群組 ID。

- 部署 ID 資料夾表示部署群組中的每一個部署。每一個部署目錄的名稱便是其 ID。每個資料夾都包含：

  - bundle.tar – 包含部署修訂內容的壓縮檔案。若要檢視修訂內容，則可使用 zip 格式的解壓縮公用程式。
  - deployment-archive – 包含部署修訂內容的目錄。
  - logs，一個包含 scripts.log 檔案的目錄。此檔案會列示部署檔案中指定的所有程序 AppSpec 檔的輸出。

  如果您想要尋找用於部署的資料夾，但不知道其部署識別碼或部署群組識別碼，可以使用AWS CodeDeploy 主控台或尋找它們。 AWS CLI 如需詳細資訊，請參閱 檢視 CodeDeploy 部署詳情 。

  部署群組中預設可封存的部署數上限為五。當到達這數字後，便會封存未來的部署，並刪除最舊的封存。您可以使用 CodeDeploy 代理程式組態檔中的 max_ 修訂設定來變更預設值。如需詳細資訊，請參閱 CodeDeploy 用戶端組態參考。

  > **Note**
  >
  > 若您想要還原封存部署所使用的硬碟空間，請更新 max_revisions 設定，將該值減少 1 或 2。下一個部署會刪除封存部署，所以這個數目等於是您指定的。

- deployment-instructions 包含每個部署群組的四個文字檔：
  - [Deployment Group ID]-cleanup，一個文字檔，具有在部署期間執行之每個命令的還原版本 undo 。範例檔案名稱為acde1916-9099-7caf-fd21-012345abcdef-cleanup
  - [Deployment Group ID]-install.json，在最新的部署期間建立的 JSON 檔。它包含在部署中執行的命令。範例檔案名稱為acde1916-9099-7caf-fd21-012345abcdef-install.json
  - [Deployment Group ID]_last_successfull_install，一個文字檔，其列出上一次成功部署的封存目錄。當代理程式已將部署應用程 CodeDeploy 式中的所有檔案複製到執行個體時，便會建立此檔案。 CodeDeploy 代理程式會在下次部署期間使用它來決定要執行的BeforeInstall指令碼ApplicationStop和指令碼。範例檔案名稱為acde1916-9099-7caf-fd21-012345abcdef_last_successfull_install

  - [Deployment Group ID]_most_recent_install，一個文字檔，其列出最近部署的封存目錄的名稱。這個檔案是在部署中的檔案成功下載時建立的。[deployment group ID]_last_successfull_install 檔案則在下載的檔案複製到其最終目的地後建立。範例檔案名稱為`acde1916-9099-7caf-fd21-012345abcdef_most_recent_install`
- deployment-logs 包含下列日誌檔：
  - 每天有部署時建立 codedeploy-agent.yyyymmdd.log 檔案。每個日誌檔案包含當日的部署資訊。這些日誌檔也許對偵測問題有用處，例如權限問題。這個日誌檔原本命名為 `codedeploy-agent.log`。隔天，其部署的日期會插入檔案名稱。例如，如果今天是 2018 年 1 月 3 日，您可以查看有關所有今日在`codedeploy-agent.log`的部署 。明日 2018，1 月 4 日，日誌檔會重新命名 `codedeploy-agent.20180103.log`。
  - codedeploy-agent-deployments.log 編譯每個部署的`scripts.log`檔案內容。`scripts.log` 檔案位於每個 `Deployment ID` 資料夾下的 `logs` 子資料夾。此檔案中的項目前面是部署 ID。例如，「`[d-ABCDEF123]LifecycleEvent - BeforeInstall`」可能會在 ID 為 `d-ABCDEF123` 的部署期間寫入。當`codedeploy-agent-deployments.log`達到其大小上限時，CodeDeploy 代理程式會在刪除舊內容時繼續寫入該代理程式。

管理 CodeDeploy 代理程式作

本節中的指示說明如何安裝、解除安裝、重新安裝或更新 CodeDeploy 代理程式，以及如何驗證 CodeDeploy 代理程式是否正在執行。

主題

確認 CodeDeploy 代理程式正在執行

本節說明如果您懷疑 CodeDeploy 代理程式已停止在執行個體上執行時要執行的命令。

主題

確認 Amazon Linux 或 RHEL 的 CodeDeploy 代理程式正在執行

若要查看 CodeDeploy 代理程式是否已安裝並執行，請登入執行個體，然後執行下列命令：

```
systemctl status codedeploy-agent
```

如果命令傳回錯誤，表示不會安裝 CodeDeploy 代理程式。按照 安裝 Amazon Linux 或 RHEL 的 CodeDeploy 代理程式 中的說明加以安裝。

如果已安裝並執行 CodeDeploy 代理程式，您應該會看到類似的訊息`The AWS CodeDeploy agent is running`。

如果您看到類似 `error: No AWS CodeDeploy agent running` 的訊息，請啟動服務並執行以下兩個命令，一次一個：

```
systemctl start codedeploy-agent
```

```
systemctl status codedeploy-agent
```

確認 Ubuntu 伺服器的 CodeDeploy 代理程式正在執行中

若要查看 CodeDeploy 代理程式是否已安裝並執行，請登入執行個體，然後執行下列命令：

```
systemctl status codedeploy-agent
```

如果命令傳回錯誤，表示不會安裝 CodeDeploy 代理程式。按照 安裝 Ubuntu 伺服器的 CodeDeploy 代理程式 中的說明加以安裝。

如果已安裝並執行 CodeDeploy 代理程式，您應該會看到類似的訊息`The AWS CodeDeploy agent is running`。

如果您看到類似 `error: No AWS CodeDeploy agent running` 的訊息，請啟動服務並執行以下兩個命令，一次一個：

```
systemctl start codedeploy-agent
```

```
systemctl status codedeploy-agent
```

確認 Windows 伺服器的 CodeDeploy 代理程式是否正在執行

若要查看 CodeDeploy 代理程式是否已安裝並執行，請登入執行個體，然後執行下列命令：

```
powershell.exe -Command Get-Service -Name codedeployagent
```

您應該會看到類似下列的輸出：

```
Status   Name               DisplayName
------   ----               -----------
Running codedeployagent    CodeDeploy Host Agent Service
```

如果命令傳回錯誤，表示不會安裝 CodeDeploy 代理程式。按照 安裝 Windows 伺服器的 CodeDeploy 代理程式 中的說明加以安裝。

如果 `Status` 顯示 `Running` 以外的服務，請使用下列命令啟動服務：

```
powershell.exe -Command Start-Service -Name codedeployagent
```

您可以按照下列命令重新啟動服務：

```
powershell.exe -Command Restart-Service -Name codedeployagent
```

您可以按照下列命令停止服務：

```
powershell.exe -Command Stop-Service -Name codedeployagent
```

判斷代 CodeDeploy理程式的版本

您可以透過兩種方式判斷執行個體上執行的 CodeDeploy 代理程式版本。

首先，從 CodeDeploy 代理程式的版本 1.0.1.854 開始，您可以在執行個體上的`.version`檔案中檢視版本號碼。下表顯示每個支援的作業系統的位置和範例版本字串。

作業系統	檔案位置	範例 agent_version 字串
Amazon Linux 和紅帽企業 Linux (RHEL)	`/opt/codedeploy-agent/.version`	OFFICIAL_1.0.1.854_rpm
Ubuntu Server	`/opt/codedeploy-agent/.version`	OFFICIAL_1.0.1.854_deb
Windows Server	`C:\ProgramData\Amazon\CodeDeploy\.version`	OFFICIAL_1.0.1.854_msi

其次，您可以在執行個體上執行命令，以判斷 CodeDeploy代理程式的版本。

主題

確定 Amazon Linux 或 RHEL 上的版本

登入執行個體並執行下列命令：

```
sudo yum info codedeploy-agent
```

確定 Ubuntu 服務器上的版本

登入執行個體並執行下列命令：

```
sudo dpkg -s codedeploy-agent
```

確定視窗伺服器上的版本

登入執行個體並執行下列命令：

```
sc qdescription codedeployagent
```

安裝 CodeDeploy 代理程式

若要 CodeDeploy 在 EC2 執行個體或現場部署伺服器上使用，必須先安裝 CodeDeploy 代理程式。我們建議使用安裝和更新 CodeDeploy 代理程式 AWS Systems Manager。如需有關 Systems Manager 的詳細資訊，請參閱什麼是 AWS Systems Manager。您可以在建立部署群組時，使用主控台中的 Systems Manager 來設定 CodeDeploy代理程式的安裝和排程更新。

您也可以使用命令列直接從 S3 儲存貯體安裝 CodeDeploy 代理程式。

如需要安裝的建議版本，請參閱 CodeDeploy代理程式的版本歷史記錄。

主題

- 使用安裝 CodeDeploy 代理程式 AWS Systems Manager
- 使用命令列安裝 CodeDeploy 代理程式

使用安裝 CodeDeploy 代理程式 AWS Systems Manager

您可以使用 AWS Management Console 或將 CodeDeploy 代理程式安裝 AWS CLI 到 Amazon EC2 或現場部署執行個體，方法是使用 AWS Systems Manager。您可以選擇安裝特定版本，或選擇永遠安裝最新版本的代理程式。如需有關的詳細資訊 AWS Systems Manager，請參閱什麼是 AWS Systems Manager。

建議使用 AWS Systems Manager 是安裝和更新 CodeDeploy 代理程式的方法。您也可以從 Amazon S3 儲存貯體安裝 CodeDeploy 代理程式。如需有關使用 Simple Storage Service (Amazon S3) 下載連結的資訊，請參閱 使用命令列安裝 CodeDeploy 代理程式。

主題

- 必要條件
- 安裝 CodeDeploy 代理程式

必要條件

按照中開始使用 CodeDeploy的步驟設定 IAM 許可和 AWS CLI.

如果使用系統管理員在現場部署伺服器上安裝 CodeDeploy 代理程式，您必須向 Amazon EC2 Systems Manager 註冊現場部署伺服器。如需詳細資訊，請參閱AWS Systems Manager 使用者指南中的在混合式環境中設定 Systems Manager。

安裝 CodeDeploy 代理程式

在您可以使用 Systems Manager 安裝 CodeDeploy 代理程式之前，您必須確定 Systems Manager 已正確設定執行個體。

安裝或更新 SSM 代理程式

在 Amazon EC2 執行個體上， CodeDeploy 代理程式要求執行個體執行的是 2.3.274.0 或更新版本。安裝 CodeDeploy 代理程式之前，請先在執行個體上更新或安裝 SSM 代理程式 (如果尚未安裝)。

SSM 代理程式已預先安裝在由提供的某些 Amazon EC2 AMI 上。 AWS如需詳細資訊，請參閱已預先安裝 SSM 代理程式的 Amazon 機器映像 (AMI)。

> **Note**
>
> 請確定 CodeDeploy代理程式也支援執行個體的作業系統。如需詳細資訊，請參閱 CodeDeploy 代理程式支援的作業系統。

如需在執行 Linux 的執行個體上安裝或更新 SSM 代理程式的詳細資訊，請參閱《AWS Systems Manager 使用指南》中的〈在 Linux 執行個體上安裝和設定 SSM 代理程式〉。

如需有關在執行 Windows Server 的執行個體上安裝或更新 SSM 代理程式的詳細資訊，請參閱《AWS Systems Manager 使用指南》中的在 Windows 執行個體上安裝和設定 SSM 代理程式。

(選用) 驗證 Systems Manager 先決條件

在您使用 Systems Manager 執行命令來安裝 CodeDeploy 代理程式之前，請確認您的執行個體符合 Systems Manager 的最低需求。若要取得更多資訊，請參閱《使用指南》 AWS Systems Manager中的AWS Systems Manager 〈設定〉。

安裝 CodeDeploy 代理程式

使用 SSM，您可以安裝一 CodeDeploy 次或設定排程來安裝新版本。

如果要安裝 CodeDeploy 代理程式，請選擇`AWSCodeDeployAgent`套件，同時遵循「使用 AWS Systems Manager 發行者安裝或更新套件」中的步驟。

使用命令列安裝 CodeDeploy 代理程式

> **Note**
>
> 建議您使用安裝 CodeDeploy 代理程式，以 AWS Systems Manager 便能夠設定代理程式的預約更新。如需詳細資訊，請參閱 使用安裝 CodeDeploy 代理程式 AWS Systems Manager。

使用下列主題可使用命令列安裝和執行 CodeDeploy 代理程式。

主題

- 安裝 Amazon Linux 或 RHEL 的 CodeDeploy 代理程式
- 安裝 Ubuntu 伺服器的 CodeDeploy 代理程式
- 安裝 Windows 伺服器的 CodeDeploy 代理程式

安裝 Amazon Linux 或 RHEL 的 CodeDeploy 代理程式

登入執行個體，並執行下列命令，一次一個：在使yum用安裝套件時，`sudo yum update`先執行指令會被視為最佳作法，但如果您不想更新所有套件，可以略過此指令。

```
sudo yum update
```

```
sudo yum install ruby
```

```
sudo yum install wget
```

(選擇性) 若要清除任何先前代理程式快取資訊的 AMI，請執行下列指令碼：

```
#!/bin/bash
CODEDEPLOY_BIN="/opt/codedeploy-agent/bin/codedeploy-agent"
$CODEDEPLOY_BIN stop
yum erase codedeploy-agent -y
```

切換到您的主目錄：

```
cd /home/ec2-user
```

> **Note**
>
> 在上一個命令中，/home/ec2-user代表 Amazon Linux 或 RHEL 亞馬 Amazon EC2 執行個體的預設使用者名稱。如果您的執行個體是使用自訂的 AMI 建立的，AMI 擁有者可能已指定不同的預設使用者名稱。

下載 CodeDeploy 代理程式安裝程式：

```
wget https://bucket-name.s3.region-identifier.amazonaws.com/latest/install
```

值區##是 Amazon S3 儲存貯體的名稱，該儲存貯體包含您區域的 CodeDeploy 資源套件檔案，而區域##########。

例如：

```
https://aws-codedeploy-us-east-2.s3.us-east-2.amazonaws.com/latest/install
```

如需值區名稱和地區識別碼的清單，請參閱依區域的資源套件時段名稱。

設置install文件的執行權限：

```
chmod +x ./install
```

如果要安裝最新版本的 CodeDeploy 代理程式：

- ```
  sudo ./install auto
  ```

如果要安裝特定版本的 CodeDeploy 代理程式：

- 列出您所在地區的可用版本：

  ```
  aws s3 ls s3://aws-codedeploy-region-identifier/releases/ --region region-identifier
   | grep '\.rpm$'
  ```

- 安裝下列其中一個版本：

  ```
  sudo ./install auto -v releases/codedeploy-agent-version.noarch.rpm
  ```
  ```

> **Note**
>
> AWS 支援 CodeDeploy 代理程式的最新次要版本。目前最新的次要版本是 1.7.x。

若要確認服務是否正在執行，請執行下列命令：

```
systemctl status codedeploy-agent
```

如果已安裝並執行 CodeDeploy 代理程式，您應該會看到類似的訊息`The AWS CodeDeploy agent is running`。

如果您看到類似 `error: No AWS CodeDeploy agent running` 的訊息，請啟動服務並執行以下兩個命令，一次一個：

```
systemctl start codedeploy-agent
```

```
systemctl status codedeploy-agent
```

安裝 Ubuntu 伺服器的 CodeDeploy 代理程式

> **Note**
>
> 建議您使用安裝 CodeDeploy 代理程式，以 AWS Systems Manager 便能夠設定代理程式的預約更新。如需詳細資訊，請參閱 使用安裝 CodeDeploy 代理程式 AWS Systems Manager。

若要在 Ubuntu 伺服器上安裝 CodeDeploy 代理程式

1. 登入執行個體。
2. 一個接一個地輸入以下命令：

   ```
   sudo apt update
   ```

   ```
   sudo apt install ruby-full
   ```

```
sudo apt install wget
```

3. 輸入以下命令：

```
cd /home/ubuntu
```

/##/ubuntu 代表一個 Ubuntu 服務器實例的默認用戶名。如果您的執行個體是使用自訂的 AMI 建立的，AMI 擁有者可能已指定不同的預設使用者名稱。

4. 輸入以下命令：

```
wget https://bucket-name.s3.region-identifier.amazonaws.com/latest/install
```

值區*##*是 Amazon S3 儲存貯體的名稱，該儲存貯體包含您區域的 CodeDeploy 資源套件檔案，而區域*##########*。

例如：

```
https://aws-codedeploy-us-east-2.s3.us-east-2.amazonaws.com/latest/
install
```

如需值區名稱和地區識別碼的清單，請參閱依區域的資源套件時段名稱。

5. 輸入以下命令：

```
chmod +x ./install
```

6. 執行以下任意一項：

- 若要在任何受支援的 Ubuntu 伺服器版本 (20.04 除外) 上安裝最新版本的 CodeDeploy 代理程式：

```
sudo ./install auto
```

- 若要在 Ubuntu 伺服器 20.04 上安裝最新版本的 CodeDeploy 代理程式：

> **Note**
>
> 將輸出寫入臨時日誌文件是一種解決方法，當我們使用 Ubuntu Server 20.04 上的`install`腳本解決已知錯誤時，應該使用該解決方法。

```
sudo ./install auto > /tmp/logfile
```

- 若要在任何受支援的 Ubuntu 伺服器版本 (20.04 除外) 上安裝特定版本的 CodeDeploy 代理程式：
  - 列出您所在地區的可用版本：

```
aws s3 ls s3://aws-codedeploy-region-identifier/releases/ --region region-identifier | grep '\.deb$'
```

  - 安裝下列其中一個版本：

```
sudo ./install auto -v releases/codedeploy-agent-###.deb
```

> **Note**
>
> AWS 支援 CodeDeploy 代理程式的最新次要版本。目前最新的次要版本是 1.7.x。

- 若要在 Ubuntu 伺服器 20.04 上安裝特定版本的 CodeDeploy 代理程式：
  - 列出您所在地區的可用版本：

```
aws s3 ls s3://aws-codedeploy-region-identifier/releases/ --region region-identifier | grep '\.deb$'
```

  - 安裝下列其中一個版本：

```
sudo ./install auto -v releases/codedeploy-agent-###.deb > /tmp/logfile
```

> **Note**
>
> 將輸出寫入臨時日誌文件是一種解決方法，當我們使用 Ubuntu Server 20.04 上的`install`腳本解決已知錯誤時，應該使用該解決方法。

> **Note**
>
> AWS 支援 CodeDeploy 代理程式的最新次要版本。目前最新的次要版本是 1.7.x。

檢查服務是否正在執行

1. 輸入以下命令：

   ```
   systemctl status codedeploy-agent
   ```

   如果已安裝並執行 CodeDeploy 代理程式，您應該會看到類似的訊息`The AWS CodeDeploy agent is running`。
2. 如果您看到類似 `error: No AWS CodeDeploy agent running` 的訊息，請啟動服務並執行以下兩個命令，一次一個：

   ```
   systemctl start codedeploy-agent
   ```

   ```
   systemctl status codedeploy-agent
   ```

安裝 Windows 伺服器的 CodeDeploy 代理程式

在 Windows Server 執行個體上，您可以使用下列其中一種方法來下載並安裝 CodeDeploy 代理程式：

- 使用 AWS Systems Manager （ 推薦 ）
- 執行一系列的視窗 PowerShell 指令。
- 選擇直接下載連結。
- 運行 Amazon S3 複製命令。

> **Note**
>
> 安裝 CodeDeploy 代理程式所在的資料夾`C:\Program Data\Amazon\CodeDeploy`。確保此路徑上沒有目錄連接或符號鏈接。

主題

- 使用 Systems Manager
- 使用視窗 PowerShell
- 使用直接連結
- 使用 Amazon S3 複製命令

使用 Systems Manager

遵循中的指示使用安裝 CodeDeploy 代理程式 AWS Systems Manager安裝 CodeDeploy代理程式。

使用視窗 PowerShell

登入執行個體，然後在 Windows 中執行下列命令 PowerShell：

1. 要求所有從網際網路下載的指令碼和組態檔案由信任的發行者簽署。如果您被提示更改執行政策，請輸入「**Y**」。

   ```
   Set-ExecutionPolicy RemoteSigned
   ```

2. 載入 AWS Tools for Windows PowerShell.

   ```
   Import-Module AWSPowerShell
   ```

3. 建立下載 CodeDeploy 代理程式安裝檔案的目錄。

   ```
   New-Item -Path "c:\temp" -ItemType "directory" -Force
   ```

4. 使用`Set-AWSCredential`和`Initialize-AWSDefaultConfiguration`命令設定 AWS 認證。如需詳細資訊，請參閱使用 PowerShell 者指南AWS 工具中的使用 AWS 認證。

5. 下載 CodeDeploy 代理程式安裝檔案。

   > **Note**
   >
   > AWS 支援 CodeDeploy 代理程式的最新次要版本。目前最新的次要版本是 1.7.x。

   如果要安裝最新版本的 CodeDeploy代理程式：

   - ```
     powershell.exe -Command Read-S3Object -BucketName bucket-name -Key latest/
     codedeploy-agent.msi -File c:\temp\codedeploy-agent.msi
     ```

   如果要安裝特定版本的 CodeDeploy代理程式：

   - ```
     powershell.exe -Command Read-S3Object -BucketName bucket-name -Key releases/
     codedeploy-agent-###.msi -File c:\temp\codedeploy-agent.msi
     ```

儲存####是 Amazon S3 儲存貯體的名稱，該儲存貯體包含您所在地區的 CodeDeploy 資源套件檔案。######### （###） ########### aws-codedeploy-us-east-2如需值區名稱清單，請參閱依區域的資源套件時段名稱。

6. 執行 CodeDeploy 代理程式安裝檔案。

```
c:\temp\codedeploy-agent.msi /quiet /l c:\temp\host-agent-install-log.txt
```

若要確認服務是否正在執行，請執行下列命令：

```
powershell.exe -Command Get-Service -Name codedeployagent
```

如果 CodeDeploy 代理程式剛剛安裝且尚未啟動，則在執行Get-Service命令之後，在「狀態」下，您應該會看到**Start...**：

```
Status     Name               DisplayName
------     ----               -----------
Start...   codedeployagent    CodeDeploy Host Agent Service
```

如果 CodeDeploy 代理程式已在執行中，則在執行Get-Service命令之後，在「狀態」下，您應該會看到**Running**：

```
Status     Name               DisplayName
------     ----               -----------
Running    codedeployagent    CodeDeploy Host Agent Service
```

使用直接連結

如果 Windows Server 執行個體上的瀏覽器安全性設定提供權限 (例如，到https://s3.*.amazonaws.com)，您可以使用區域的直接連結下載 CodeDeploy 代理程式，然後手動執行安裝程式。

鏈接是：

```
https://s3.region.amazonaws.com/aws-codedeploy-region/latest/codedeploy-agent.msi
```

... *##*是您要部署應用程式的 AWS 區域。

例如：

```
https://s3.af-south-1.amazonaws.com/aws-codedeploy-af-south-1/latest/codedeploy-
agent.msi
```

> **Important**
>
> 從與您的 CodeDeploy應用程式相同的區域取得`.msi`檔案。當您執行`codedeploy-agent-log`檔案時，選擇不同的區域可能會導致`.msi`檔案`inconsistent region`失敗。

使用 Amazon S3 複製命令

如果執行 AWS CLI 個體上已安裝，您可以使用 Amazon S3 cp 命令下載 CodeDeploy 代理程式，然後手動執行安裝程式。如需詳細資訊，請參閱AWS Command Line Interface 在 Microsoft 視窗上安裝。

Amazon S3 命令是：

```
aws s3 cp s3://aws-codedeploy-region/latest/codedeploy-agent.msi codedeploy-agent.msi
 --region region
```

... *##*是您要部署應用程式的 AWS 區域。

例如：

```
aws s3 cp s3://aws-codedeploy-af-south-1/latest/codedeploy-agent.msi codedeploy-
agent.msi --region af-south-1
```

更新 CodeDeploy 代理程式

您可以使用，在所有支援的作業系統上設定 CodeDeploy 代理程式的自動預約更新 AWS Systems Manager。您也可以藉由在執行個體上執行命令，強制在所有支援的作業系統進行更新。

主題

- 在 Amazon Linux 或 RHEL 上更新 CodeDeploy 代理程式
- 更新 Ubuntu 伺服器上的 CodeDeploy 代理程式
- 更新 Windows 伺服器上的 CodeDeploy 代理程式

在 Amazon Linux 或 RHEL 上更新 CodeDeploy 代理程式

如果要使用設定 CodeDeploy 代理程式的自動預約更新 AWS Systems Manager，請依照使用安裝 CodeDeploy 代理程式中的步驟執行 AWS Systems Manager。

如果您要強制更新 CodeDeploy 代理程式，請登入執行個體，然後執行下列命令：

```
sudo /opt/codedeploy-agent/bin/install auto
```

更新 Ubuntu 伺服器上的 CodeDeploy 代理程式

如果要使用設定 CodeDeploy 代理程式的自動預約更新 AWS Systems Manager，請依照使用安裝 CodeDeploy 代理程式中的步驟執行 AWS Systems Manager。

如果您要強制更新 CodeDeploy 代理程式，請登入執行個體，然後執行下列命令：

```
sudo /opt/codedeploy-agent/bin/install auto
```

更新 Windows 伺服器上的 CodeDeploy 代理程式

您可以使用啟用 CodeDeploy 代理程式的自動更新 AWS Systems Manager。使用 Systems Manager，您可以透過與系統管理員狀態管理員建立關聯，為 Amazon EC2 或現場部署執行個體設定更新排程。您也可以解除安裝目前版本並安裝較新的 CodeDeploy 代理程式，以手動更新代理程式。

主題

- 設定自動 CodeDeploy代理程式更新 AWS Systems Manager
- 手動更新 CodeDeploy 代理程式
- (已取代) 使用 Windows 伺服器更新程式更新 CodeDeploy 代理程式

設定自動 CodeDeploy代理程式更新 AWS Systems Manager

若要設定 Systems Manager 並啟用 CodeDeploy 代理程式的自動更新，請遵循使用安裝 CodeDeploy 代理程式中的指示 AWS Systems Manager。

手動更新 CodeDeploy 代理程式

如果要手動更新 CodeDeploy 代理程式，您可以從 CLI 或使用 Systems Manager 安裝最新版本。遵循安裝 CodeDeploy代理程式中的指示。建議您依照解除安裝 CodeDeploy 代理程式中的指示，解除安裝舊版 CodeDeploy代理程式。

(已取代) 使用 Windows 伺服器更新程式更新 CodeDeploy 代理程式

> **Note**
>
> Windows 伺服器的 CodeDeploy 代理程式更新程式已取代，且不會更新至 1.1.1597 之後的任何版本。

如果要啟用 CodeDeploy 代理程式的自動更新，請在新的或現有的執行個體上安裝 Windows Server 的 CodeDeploy 代理程式更新程式。定期檢查的更新程式的新版本。當偵測到新版本時，更新程式會移除目前的代理程式版本 (若有安裝的話)，然後再安裝最新版本。

更新程式偵測到新版本時，如果部署已在進行，部署會繼續完成。如果部署嘗試在更新程序期間啟動，部署會失敗。

如果您要強制更新 CodeDeploy 代理程式，請遵循中的指示安裝 Windows 伺服器的 CodeDeploy 代理程式。

在 Windows 伺服器執行個體上，您可以透過執行 Windows 命 PowerShell 令、使用直接下載連結或執行 Amazon S3 複製命令來下載和安裝 CodeDeploy 代理程式更新程式。

主題

- 使用視窗 PowerShell
- 使用直接連結
- 使用 Amazon S3 複製命令

使用視窗 PowerShell

登入執行個體，並在 Windows PowerShell 中一次執行一個下列命令：

```
Set-ExecutionPolicy RemoteSigned
```

如果系統提示您變更執行原則，請選擇 **Y** Windows 會 PowerShell 要求從網際網路下載的所有指令碼和組態檔案都必須由受信任的發行者簽署。

```
Import-Module AWSPowerShell
```

```
New-Item -Path "c:\temp" -ItemType "directory" -Force
```

```
powershell.exe -Command Read-S3Object -BucketName bucket-name -Key latest/codedeploy-
agent-updater.msi -File c:\temp\codedeploy-agent-updater.msi
```

```
c:\temp\codedeploy-agent-updater.msi /quiet /l c:\temp\host-agent-updater-log.txt
```

```
powershell.exe -Command Get-Service -Name codedeployagent
```

儲存####是 Amazon S3 儲存貯體的名稱，該儲存貯體包含您所在地區的 CodeDeploy 資源套件檔案。######### （###） ############ aws-codedeploy-us-east-2如需值區名稱清單，請參閱依區域的資源套件時段名稱。

如果您需要疑難排解更新程序錯誤，請輸入下列命令以開啟 CodeDeploy 代理程式更新程式記錄檔：

```
notepad C:\ProgramData\Amazon\CodeDeployUpdater\log\codedeploy-agent.updater.log
```

使用直接連結

如果 Windows Server 執行個體上的瀏覽器安全性設定提供必要的權限 (例如，到 `http://s3.*.amazonaws.com`)，您可以使用直接連結來下載 CodeDeploy 代理程式更新程式。

鏈接是：

```
https://s3.region.amazonaws.com/aws-codedeploy-region/latest/codedeploy-agent-
updater.msi
```

... 地#是您要更新應用程式的 AWS 地區。

例如：

```
https://s3.af-south-1.amazonaws.com/aws-codedeploy-af-south-1/latest/codedeploy-agent-
updater.msi
```

使用 Amazon S3 複製命令

如果執行 AWS CLI 個體上已安裝，您可以使用 Amazon S3 cp 命令下載 CodeDeploy 代理程式更新程式，然後手動執行安裝程式。如需詳細資訊，請參閱AWS Command Line Interface 在 Microsoft 視窗上安裝。

Amazon S3 命令是：

```
aws s3 cp s3://aws-codedeploy-region/latest/codedeploy-agent-updater.msi codedeploy-
agent-updater.msi --region region
```

... 地#是您要更新應用程式的 AWS 地區。

例如：

```
aws s3 cp s3://aws-codedeploy-af-south-1/latest/codedeploy-agent-updater.msi
 codedeploy-agent-updater.msi --region af-south-1
```

解除安裝 CodeDeploy 代理

當 CodeDeploy 代理程式不再需要時，或想要執行全新安裝時，您可以從執行個體中移除代理程式。

從 Amazon Linux 或 RHEL 解除安裝 CodeDeploy 代理程式

若要解除安裝 CodeDeploy 代理程式，請登入執行個體並執行下列命令：

```
sudo yum erase codedeploy-agent
```

從 Ubuntu 伺服器中解除安裝 CodeDeploy 代理

若要解除安裝 CodeDeploy 代理程式，請登入執行個體並執行下列命令：

```
sudo dpkg --purge codedeploy-agent
```

從 Windows 伺服器解除安裝 CodeDeploy 代理程式

若要解除安裝 CodeDeploy 代理程式，請登入執行個體並執行下列三個命令，一次執行一個命令：

```
wmic
```

```
product where name="CodeDeploy Host Agent" call uninstall /nointeractive
```

```
exit
```

您也可以登入執行個體，然後在 [控制台] 中開啟 [程式和功能]，選擇 [CodeDeploy 主機代理程式]，然後選擇 [解除安裝]。

傳送 CodeDeploy 代理程式記錄至 CloudWatch

您可以 CloudWatch 使用統一的 CodeDeploy 代理程式，或者更簡單地說，將 CloudWatch 代理程式指標和記錄資料傳送至 CloudWatch 代理程式。

請遵循下列指示來安裝 CloudWatch 代理程式，並將其設定為與 CodeDeploy代理程式搭配使用。

必要條件

開始之前，請先完成以下任務：

- 安裝 CodeDeploy 代理程式並確定代理程式正在執行。如需詳細資訊，請參閱 安裝 CodeDeploy 代理程式 及 確認 CodeDeploy 代理程式正在執行。
- 安裝代 CloudWatch 理程式。如需詳細資訊，請參閱安裝 CloudWatch 代理程式。
- 將下列許可新增至 CodeDeploy IAM 執行個體設定檔：
  - CloudWatchLogsFullAccess
  - CloudWatchAgentServerPolicy

  如需 CodeDeploy 執行個體設定檔的詳細資訊，請參閱中步驟 4：為您的 Amazon EC2 執行個體建立 IAM 執行個體設定檔的開始使用 CodeDeploy。

設定 CloudWatch 代理程式以收集 CodeDeploy 記錄檔

您可以透過逐步執行精靈或手動建立或編輯組態檔來設定 CloudWatch 代理程式。

使用精靈設定 CloudWatch 代理程式 (Linux)

1. 執行精靈，如執行 CloudWatch 代理程式組態精靈中所述。
2. 在嚮導中，當被問及`Do you want to monitor any log files?`輸入**1**。
3. 指定 CodeDeploy 代理程式記錄檔，執行方式如下：
   1. `Log file path`輸入記 CodeDeploy 錄檔的路徑，例如：**`/var/log/aws/codedeploy-agent/codedeploy-agent.log`**。
   2. 針對`Log group name`輸入記錄群組名稱，例如：**`codedeploy-agent-log`**。

3. 若要Log stream name輸入記錄資料流名稱，例如：**{instance_id}-codedeploy-agent-log**。

4. 當系統詢問時Do you want to specify any additional log files?，請輸入**1**。

5. 指定 CodeDeploy 代理程式部署記錄檔，執行方式如下：

   1. 若要Log file path輸入 CodeDeploy 部署記錄檔的路徑，例如：**/opt/codedeploy-agent/deployment-root/deployment-logs/codedeploy-agent-deployments.log**
   2. 針對Log group name輸入記錄群組名稱，例如：**codedeploy-agent-deployment-log**。
   3. 若要Log stream name輸入記錄資料流名稱，例如：**{instance_id}-codedeploy-agent-deployment-log**。

6. 當系統詢問時Do you want to specify any additional log files?，請輸入**1**。

7. 指定 CodeDeploy 代理程式更新程式記錄檔，執行方式如下：

   1. Log file path輸入 CodeDeploy 更新程式記錄檔的路徑，例如：**/tmp/codedeploy-agent.update.log**。
   2. 針對Log group name輸入記錄群組名稱，例如：**codedeploy-agent-updater-log**。
   3. 若要Log stream name輸入記錄資料流名稱，例如：**{instance_id}-codedeploy-agent-updater-log**。

使用精靈設定 CloudWatch 代理程式 (Windows)

1. 執行精靈，如執行 CloudWatch 代理程式組態精靈中所述。

2. 在嚮導中，當被問及Do you want to monitor any customized log files?輸入**1**。

3. 指定記 CodeDeploy 錄檔，執行方式如下：

   1. 如需Log file path輸入 CodeDeploy 代理程式記錄檔的路徑 r，例如：**C:\ProgramData\Amazon\CodeDeploy\log\codedeploy-agent-log.txt**。
   2. 針對Log group name輸入記錄群組名稱，例如：**codedeploy-agent-log**。
   3. 若要Log stream name輸入記錄資料流名稱，例如：**{instance_id}-codedeploy-agent-log**。

4. 當系統詢問時Do you want to specify any additional log files?，請輸入**1**。

5. 指定 CodeDeploy 代理程式部署記錄檔，執行方式如下：

1. 若要Log file path輸入 CodeDeploy 部署記錄檔的路徑，例如：**C:\ProgramData\Amazon\CodeDeploy\deployment-logs\codedeploy-agent-deployments.log**。
2. 針對Log group name輸入記錄群組名稱，例如：**codedeploy-agent-deployment-log**。
3. 若要Log stream name輸入記錄資料流名稱，例如：**{instance_id}-codedeploy-agent-deployment-log**。

透過手動建立或編輯組態檔來設定 CloudWatch 代理程式 (Linux)

1. 依照手動建立或編輯 CloudWatch 代理程式組態檔中所述，建立或編輯 CloudWatch 代理程式組態檔案。
2. 請確定已呼叫檔案，/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json且其中包含下列程式碼：

```
...
"logs": {
    "logs_collected": {
        "files": {
            "collect_list": [
                {
                    "file_path": "/var/log/aws/codedeploy-agent/codedeploy-
agent.log",
                    "log_group_name": "codedeploy-agent-log",
                    "log_stream_name": "{instance_id}-agent-log"
                },
                {
                    "file_path": "/opt/codedeploy-agent/deployment-root/deployment-
logs/codedeploy-agent-deployments.log",
                    "log_group_name": "codedeploy-agent-deployment-log",
                    "log_stream_name": "{instance_id}-codedeploy-agent-deployment-
log"
                },
                {
                    "file_path": "/tmp/codedeploy-agent.update.log",
                    "log_group_name": "codedeploy-agent-updater-log",
                    "log_stream_name": "{instance_id}-codedeploy-agent-updater-log"
                }
            ]
        }
```

```
    }
}
...
```

透過手動建立或編輯組態檔來設定 CloudWatch 代理程式 (Windows)

1. 依照手動建立或編輯 CloudWatch 代理程式組態檔中所述，建立或編輯 CloudWatch 代理程式組態檔案。
2. 請確定已呼叫檔案，`C:\ProgramData\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent.json`且其中包含下列程式碼：

```
...
"logs": {
        "logs_collected": {
            "files": {
                "collect_list": [
                    {
                        "file_path": "C:\\ProgramData\\Amazon\\CodeDeploy\\log\
\codedeploy-agent-log.txt",
                        "log_group_name": "codedeploy-agent-log",
                        "log_stream_name": "{instance_id}-codedeploy-agent-log"
                    },
                    {
                        "file_path": "C:\\ProgramData\\Amazon\\CodeDeploy\
\deployment-logs\\codedeploy-agent-deployments.log",
                        "log_group_name": "codedeploy-agent-deployment-log",
                        "log_stream_name": "{instance_id}-codedeploy-agent-
deployment-log"
                    }
                ]
            },
            ...
        }
    },
...
```

重新啟動 CloudWatch代理

進行變更後，請依照啟動 CloudWatch 代理程式中所述重新啟動 CloudWatch 代理程式。

使用的例證 CodeDeploy

CodeDeploy 支援部署到執行 Amazon Linux、Ubuntu 伺服器、紅帽企業 Linux (RHEL) 和視窗伺服器的執行個體。

您可以用 CodeDeploy 來部署到 Amazon EC2 執行個體和現場部署執行個體。現場部署執行個體是任何非 Amazon EC2 執行個體的實體裝置，可以執行 CodeDeploy 代理程式並連線到公有 AWS 服務端點。您可以使用 CodeDeploy 將應用程式同時部署到雲端中的 Amazon EC2 執行個體，以及將應用程式部署到辦公室中的桌上型電腦或自己資料中心的伺服器。

比較 Amazon EC2 執行個體與現場部署執行

下表比較 Amazon EC2 執行個體和現場部署執行個體：

Subject	Amazon EC2 執行個體	內部部署執
您必須安裝並執行與執行個體上執行之作業系統相容的 CodeDeploy 代理程式版本。	是	是
需要執行個體才能連線到 CodeDeploy。	是	是
需要將 IAM 執行個體設定檔附加至執行個體。IAM 執行個體設定檔必須具有參與 CodeDeploy 部署的許可。如需相關資訊，請參閱步驟 4：為您的 Amazon EC2 執行個體建立 IAM 執行個體設定檔。	是	否
需要您執行下列其中一項來驗證和註冊執行個體： • 建立 IAM 角色，該角色可由 IAM 使用者在每個執行個體上假設，以擷取透過產	否	是

Subject	Amazon EC2 執行個體	內部部署執
生的定期重新整理臨時登入資料 AWS Security Token Service。 • 為每個執行個體建立 IAM 使用者，並在執行個體上以純文字形式儲存 IAM 使用者的帳戶登入資料。		
要求您必須先註冊每個執行個體， CodeDeploy 才能部署至執行個體。	否	是
要求您先標記每個執行個體，才 CodeDeploy 能部署至執行個體。	是	是
可以在 CodeDeploy 部署過程中參與 Amazon EC2 Auto Scaling 和 Elastic Load Balancing 案例。	是	否
可以從 Amazon S3 儲存貯體和 GitHub 儲存庫部署。	是	是
可以支援觸發，以在部署或執行個體中發生指定的事件時，提示傳送 SMS 或電子郵件通知。	是	是
遵守相關聯部署的計費。	否	是

的執行個體工作 CodeDeploy

若要啟動或設定執行個體以用於部署，請選擇下列說明：

我想啟動一個新的 Amazon Linux 或視窗服務器 Amazon EC2 實例。	若要以最少的精力啟動 Amazon EC2 執行個體，請參閱為 CodeDeploy (AWS CloudFormation 範本) 建立 Amazon EC2 執行個體。 若要主要由您自行啟動 Amazon EC2 執行個體，請參閱為 CodeDeploy （AWS CLI 或 Amazon EC2 控制台）創建一個 Amazon EC2 實例。
我想啟動一個新的 Ubuntu 服務器或 RHEL Amazon EC2 實例。	請參閱為 CodeDeploy （AWS CLI 或 Amazon EC2 控制台）創建一個 Amazon EC2 實例。
我想配置一個 Amazon Linux，視窗服務器，Ubuntu 服務器或 RHEL Amazon EC2 實例。	請參閱設定要使用的 Amazon EC2 執行個體 CodeDeploy。
我想要設定 Windows 伺服器、Ubuntu 伺服器或 RHEL 現場部署執行個體 (不是 Amazon EC2 執行個體的實體裝置)。	請參閱Working with On-Premises Instances。
我想 CodeDeploy 要在藍/綠部署期間佈建替換執行個體叢集。	請參閱使用中的部署 CodeDeploy。

若要在 Amazon EC2 Auto Scaling 群組中準備 Amazon EC2 執行個體，您必須遵循一些其他步驟。如需詳細資訊，請參閱 CodeDeploy 與 Amazon EC2 Auto Scaling 集成。

主題

- Tagging Instances for Deployments
- Working with Amazon EC2 Instances
- Working with On-Premises Instances
- View Instance Details
- Instance Health

標記部署群組的執行個體 CodeDeploy

若要協助管理 Amazon EC2 執行個體和現場部署執行個體，您可以使用標籤將自己的中繼資料指派給每個資源。標籤可讓您以不同的方式分類您的執行個體，(例如依據目的、擁有者或環境)。當您有許多執行個體時，這很實用。基於指派的標籤，您可以快速鑑別一個執行個體或一群執行個體之功用。每個標籤皆包含由您定義的一個金鑰與一個選用值。如需詳細資訊，請參閱標記您的 Amazon EC2 資源。

若要指定 CodeDeploy 部署群組中包含哪些執行個體，請在一或多個標記群組中指定標記。符合您標籤準則的執行個體，是該部署群組的部署建立時，安裝最新版應用程式修訂的那些。

> **Note**
>
> 您也可以在部署群組中包含 Amazon EC2 Auto Scaling 群組，但是會以名稱識別，而不是以套用至執行個體的標籤來識別。如需相關資訊，請參閱 CodeDeploy 與 Amazon EC2 Auto Scaling 集成。

部署群組中執行個體的條件可以跟單一標籤群組裡的單一標籤一樣簡單。也可以最多有三個標籤群組，每個標籤群組最多 10 個標籤。

若您使用單一標籤群組，任何可透過群組中至少一個標籤識別的執行個體都會包含在部署群組中。若您使用多個標籤群組，僅由包含 每一個 標籤群組中辨識出至少一個標籤的執行個體。

以下範例說明如何使用標籤和標籤群組可用來選擇執行個體部署群組。

主題

範例 1：單一標籤群組、單一標籤

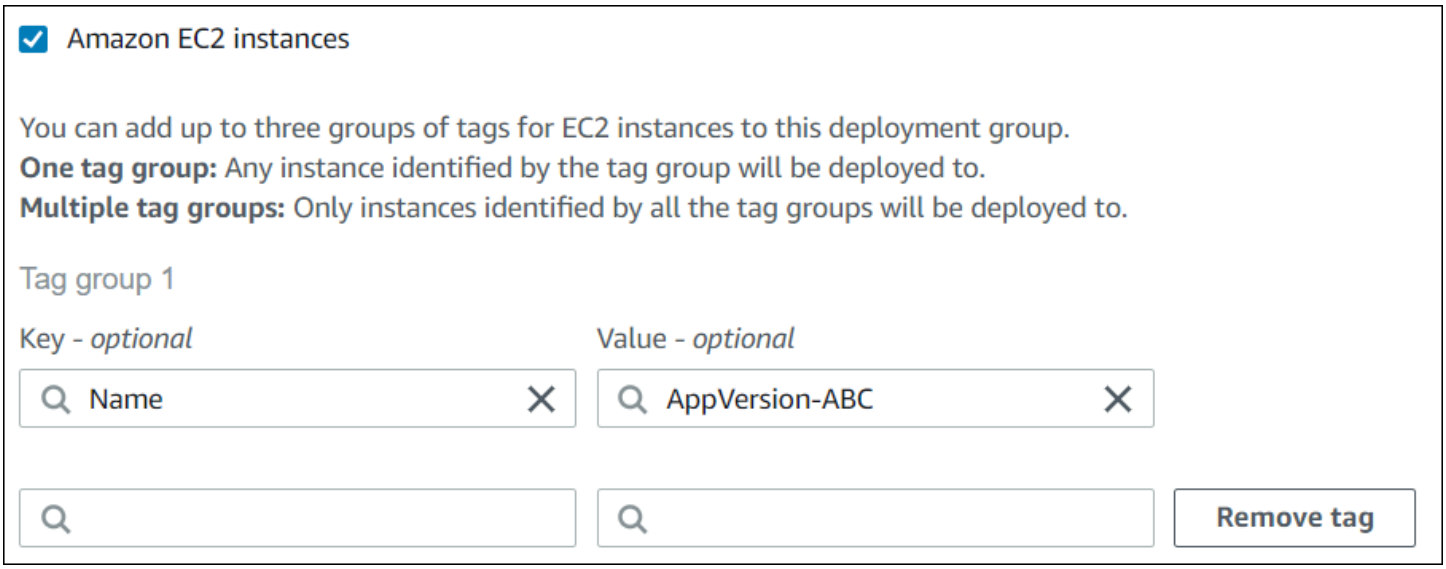
您可以指定單一標籤在單一標籤群組：

標籤群組 1

金鑰	值
名稱	AppVersion-ABC

每個使用 Name=AppVersion-ABC 標籤的執行個體，是部署群組的一部分，即使其已套用其他標籤了。

CodeDeploy 控制台設置視圖：



JSON 結構：

```
"ec2TagSet": {
         "ec2TagSetList": [
            [
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Name",
                  "Value": "AppVersion-ABC"
               }
            ]
         ]
      },
```

範例 2：單一標記群組、多個標籤

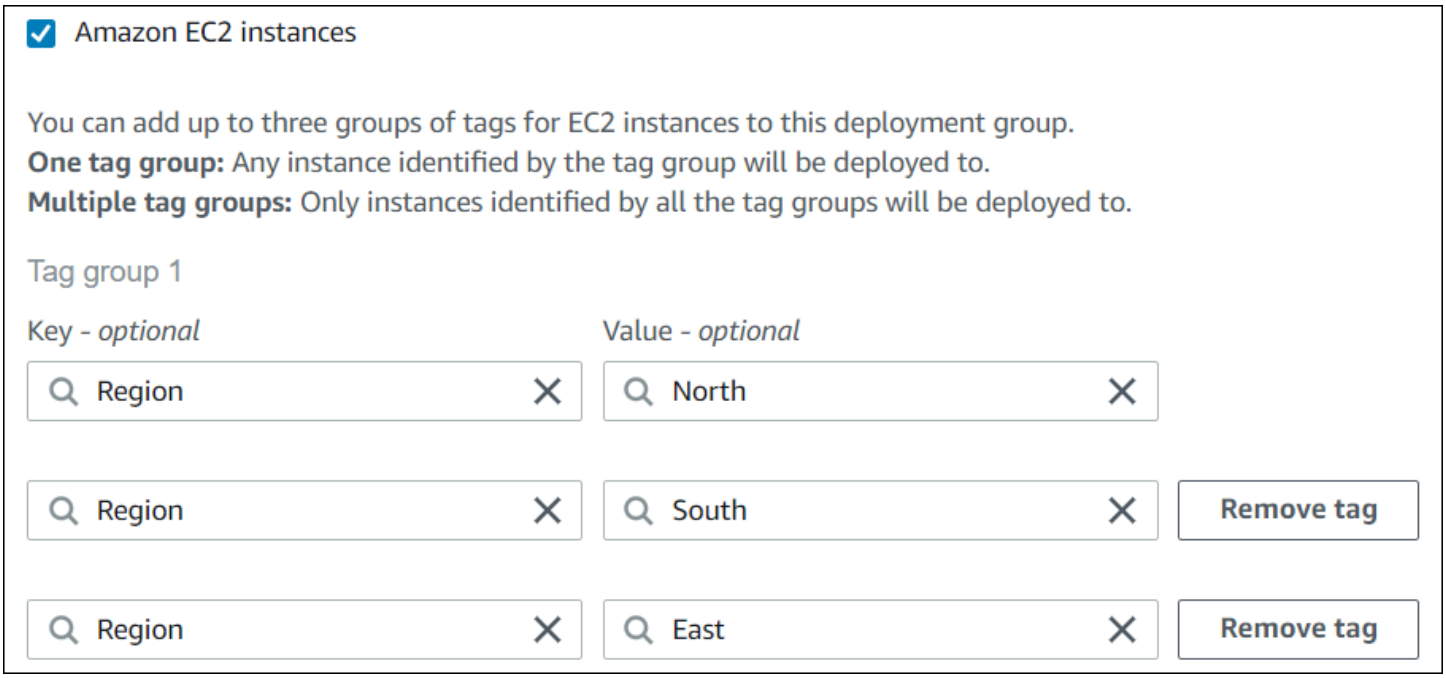
您可以指定多個標籤在單一標籤群組：

標籤群組 1

金鑰	值
區域	北
區域	南
區域	東

一個執行個體與部屬群組的某部分被標籤，即使如果它有其他標籤已套用。例如，如果您有其他 `Region=West` 標籤的執行個體，他們仍不包含在此部署群組。

CodeDeploy 控制台設置視圖：



JSON 結構：

```
"ec2TagSet": {
         "ec2TagSetList": [
            [
               {
```

```
                "Type": "KEY_AND_VALUE",
                "Key": "Region",
                "Value": "North"
            },
                        {
                "Type": "KEY_AND_VALUE",
                "Key": "Region",
                "Value": "South"
            },
                        {
                "Type": "KEY_AND_VALUE",
                "Key": "Region",
                "Value": "East"
            }
        ]
    ]
},
```

範例 3：多個標記群組、單一標記

您也可以使用有單一金鑰值對的標籤群組之多個組合，在每一個中為部署群組的執行個體指定標準。當您在部署群組中使用多個標籤群組，只有所有標籤群組識別出的執行個體會包含在部署群組中。

標籤群組 1

金鑰	值
名稱	AppVersion-ABC

標籤群組 2

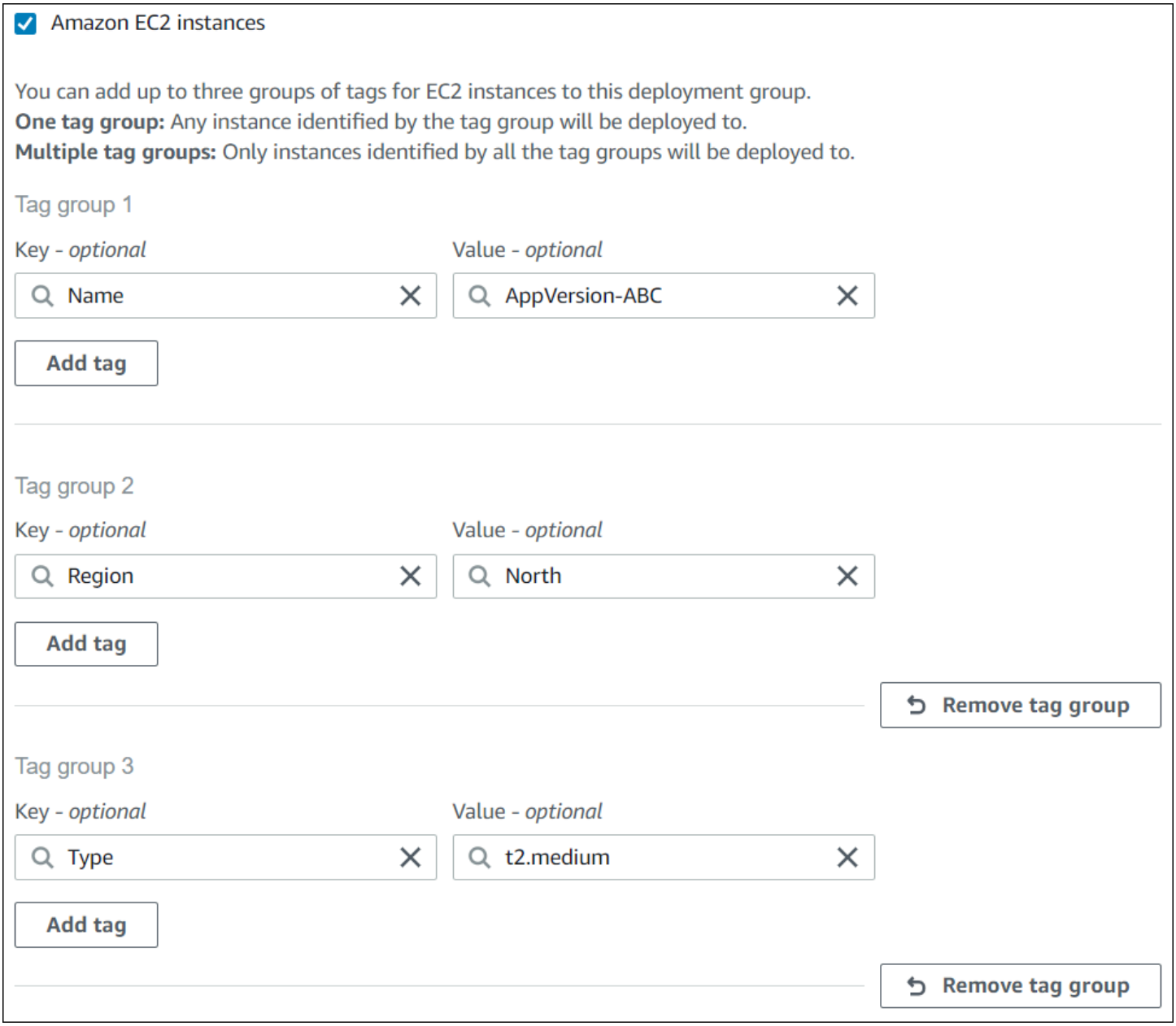
金鑰	值
區域	北

標籤群組 3

金鑰	值
Type	t2.medium

您可能在許多區域有執行個體，和許多有 Name=AppVersion-ABC 標籤的執行個體類型。在此範例中，只有含有 Region=North 和 Type=t2.medium 的執行個體是部署群組的一部分。

CodeDeploy 控制台設置視圖：



JSON 結構：

```
"ec2TagSet": {
         "ec2TagSetList": [
            [
               {
                  "Type": "KEY_AND_VALUE",
```

```
                "Key": "Name",
                "Value": "AppVersion-ABC"
            }
        ],
        [
            {
                "Type": "KEY_AND_VALUE",
                "Key": "Region",
                "Value": "North"
            }
        ],
        [
            {
                "Type": "KEY_AND_VALUE",
                "Key": "Type",
                "Value": "t2.medium"
            }
        ],
    ]
},
```

範例 4：多個標記群組、多個標籤

當您使用多個標籤群組與一個或多個群組中的多個標籤，執行個體必須與每個群組至少一個標籤相符。

標籤群組 1

金鑰	值
環境	試用版
環境	安裝

標籤群組 2

金鑰	值
區域	北
區域	南
區域	東

標籤群組 3

金鑰	值
Type	t2.medium
類型	t2.large

在這個範例中，被包含在這個部署群組裡，一個執行個體必須標記為與 (1) Environment=Beta 或 Environment=Staging，與 (2) Region=North、Region=South 或 Region=East，以及與 (3)Type=t2.medium 或 Type=t2.large 一起。

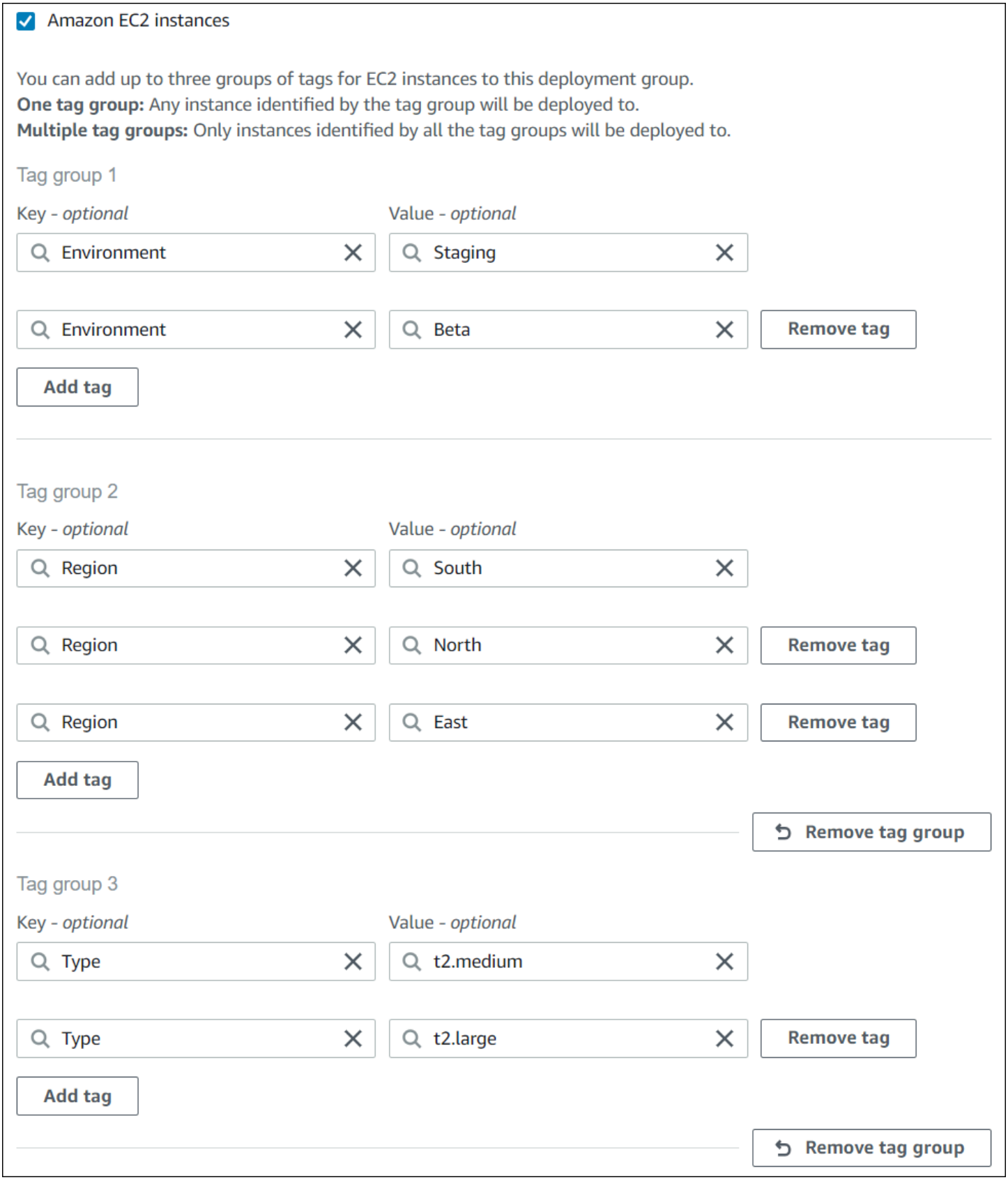
以說明，執行個體與以下標籤群組將在那些被包含在部署群組裡。

- Environment=Beta, Region=North,Type=t2.medium
- Environment=Staging,Region=East,Type=t2.large
- Environment=Staging,Region=South,Type=t2.large

執行個體與以下標籤群組將不在那些被包含在部署群組裡。醒目提示 金鑰值導致執行個體被排除在外。

- Environment=Beta、區域=西、Type=t2.medium
- Environment=Staging、Region=East、類型=t2.micro
- 環境=生產、Region=South、Type=t2.large

CodeDeploy 控制台設置視圖：


Amazon EC2 instances
You can add up to three groups of tags for EC2 instances to this deployment group.
One tag group: Any instance identified by the tag group will be deployed to.
Multiple tag groups: Only instances identified by all the tag groups will be deployed to.
Tag group 1
Key - optional
Value - optional
Environment
Staging
Environment
Beta
Remove tag
Add tag
Tag group 2
Key - optional
Value - optional
Region
South
Region
North
Remove tag
Region
East
Remove tag
Add tag
Remove tag group
Tag group 3
Key - optional
Value - optional
Type
t2.medium
Type
t2.large
Remove tag
Add tag
Remove tag group

JSON 結構：

```
"ec2TagSet": {
         "ec2TagSetList": [
            [
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Environment",
                  "Value": "Beta"
               },
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Environment",
                  "Value": "Staging"
               }
            ],
            [
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Region",
                  "Value": "North"
               },
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Region",
                  "Value": "South"
               },
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Region",
                  "Value": "East"
               }
            ],
            [
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Type",
                  "Value": "t2.medium"
               },
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Type",
                  "Value": "t2.large"
               }
```

```
            ],
        ]
    },
```

使用 Amazon EC2 執行個體 CodeDeploy

Amazon EC2 執行個體是您使用 Amazon 彈性運算雲端建立和設定的虛擬運算環境。Amazon EC2 在 AWS 雲端提供可擴展的運算容量。您可以使用 Amazon EC2 啟動 CodeDeploy 部署所需數量的虛擬伺服器。

如需有關 Amazon EC2 的詳細資訊，請參閱 Amazon EC2 入門指南。

本節中的指示說明如何建立和設定 Amazon EC2 執行個體，以便在 CodeDeploy 部署中使用。

主題

- 為 CodeDeploy （ AWS CLI 或 Amazon EC2 控制台 ）創建一個 Amazon EC2 實例
- 為 CodeDeploy (AWS CloudFormation 範本) 建立 Amazon EC2 執行個體
- 設定要使用的 Amazon EC2 執行個體 CodeDeploy

為 CodeDeploy （ AWS CLI 或 Amazon EC2 控制台 ）創建一個 Amazon EC2 實例

這些指示說明如何啟動設定用於 CodeDeploy 部署的新 Amazon EC2 執行個體。

您可以使用我們的 AWS CloudFormation 範本來啟動執行已設定用於 CodeDeploy 部署的亞馬 Amazon EC2 執行個體，或執行 Amazon Linux 或 Windows 伺服器。我們不會為執行 Ubuntu 伺服器或 RHEL (RHEL) 的 Amazon EC2 執行個體提供 AWS CloudFormation 範本。如需使用範本之替代方案的詳細資訊，請參閱使用的例證 CodeDeploy。

您可以使用 Amazon EC2 主控 AWS CLI台或 Amazon EC2 API 來啟動 Amazon EC2 執行個體。

啟動亞 Amazon EC2 實例（控制台）

必要條件

如果您尚未這樣做，請按照中的說明設開始使用 CodeDeploy定和設定 AWS CLI 和建立 IAM 執行個體設定檔。

啟動 Amazon EC2 執行個體

1. 登入 AWS Management Console 並開啟 Amazon EC2 主控台，網址為 https://console.aws.amazon.com/ec2/。
2. 在導覽窗格中，選擇 Instances (執行個體)，然後選擇 Launch Instance (啟動執行個體)。
3. 在 Step 1: Choose an Amazon Machine Image (AMI) (步驟 1：選擇 Amazon Machine Image (AMI)) 頁面上，從 Quick Start (快速入門) 標籤中找出您想要使用的作業系統和版本，然後選擇 Select (選取)。您必須選擇支援的 Amazon EC2 AMI 作業系統 CodeDeploy。如需詳細資訊，請參閱 CodeDeploy 代理程式支援的作業系統。
4. 在步驟 2：選擇執行個體類型頁面上，選擇任何可用的 Amazon EC2 執行個體類型，然後選擇下一步：設定執行個體詳細資訊。
5. 在步驟 3：設定執行個體詳細資訊頁面的 IAM 角色清單中，選擇您在其中建立的 IAM 執行個體角色步驟 4：為您的 Amazon EC2 執行個體建立 IAM 執行個體設定檔。如果您已使用建議的角色名稱，則選擇 **CodeDeployDemo-EC2-Instance-Profile**。如果您已建立自己的角色名稱，則請選取該名稱。

   > **Note**
   >
   > 如果「網路」清單中未顯示預設虛擬私有雲端 (VPC)，您必須選擇或建立 Amazon VPC 和子網路。選擇 Create new VPC (建立新 VPC) 或 Create new subnet (建立新的子網路)，或兩者皆選。如需詳細資訊，請參閱您的 VPC 和子網路。

6. 選擇 Next: Add Storage (下一步：新增儲存體)。
7. 將 Step 4: Add Storage (步驟 4：新增儲存) 頁面保持不變，然後選擇 Next: Add Tags (下一步：新增標籤)。
8. 在 Step 5: Add Tags (步驟 5：新增標籤) 頁面上，選擇 Add Tag (新增標籤)。
9. 在 Key (金鑰) 方塊中，輸入 **Name**。在 Value (值) 方塊中，輸入 **CodeDeployDemo**。

   > **Important**
   >
   > Key (金鑰) 和 Value (值) 方塊的內容區分大小寫。

10. 選擇 Next: Configure Security Group (下一步：設定安全群組)。
11. 在 Step 6: Configure Security Group (步驟 6：設定安全群組) 頁面上，選取 Create a new security group (建立新的安全群組) 選項。

為執行 Amazon Linux、Ubuntu 伺服器或 RHEL 的亞馬 Amazon EC2 執行個體設定了預設安全殼層角色。為執行 Windows 伺服器的 Amazon EC2 執行個體設定了預設 RDP 角色。

12. 如果您想要開啟 HTTP 連接埠，請選擇 Add Rule (新增規則) 按鈕，然後從 Type (類型) 下拉式清單中選擇 **HTTP**。接受預設 Source (來源) 值 Custom 0.0.0.0/0，然後選擇 Review and Launch (檢閱並啟動)。

> **Note**
>
> 在生產環境中，我們建議您限制對 SSH、RDP 和 HTTP 連接埠的存取，而不是指定任何位置 0.0.0.0/0。 CodeDeploy 不需要不受限制的連接埠存取，也不需要 HTTP 存取。如需詳細資訊，請參閱保護 Amazon EC2 執行個體的秘訣。

如果出現 Boot from General Purpose (SSD) (以一般用途 (SSD) 開機) 對話方塊，則請遵循說明，然後選擇 Next (下一步)。

13. 將 Step 7: Review Instance Launch (步驟 7：檢閱執行個體啟動) 頁面保持不變，然後選擇 Launch (啟動)。
14. 在 Select an existing key pair or create a new key pair (選取現有金鑰對或建立新的金鑰對) 對話方塊中，選擇 Choose an existing key pair (選擇現有的金鑰對) 或 Create a new key pair (建立新的金鑰對)。若您已經設定 Amazon EC2 執行個體金鑰對，則可以在這裡選擇它。

    如果您還沒有 Amazon EC2 執行個體金鑰對，則請選擇 Create a new key pair (建立新的金鑰對)，然後給予可輕鬆辨識的名稱。選擇下載 key pair，將 Amazon EC2 執行個體金鑰配對下載到您的電腦。

> **Important**
>
> 如果您想要使用安全殼層或 RDP 存取 Amazon EC2 執行個體，則必須擁有 key pair。

15. 選擇 Launch Instances (啟動執行個體)。
16. 選擇您的亞馬遜 EC2 執行個體的 ID。在啟動執行個體並通過所有檢查之前，請不要繼續進行。

安裝 CodeDeploy 代理程式

在 CodeDeploy部署中使用 CodeDeploy 代理程式之前，必須先在 Amazon EC2 執行個體上安裝代理程式。如需詳細資訊，請參閱 安裝 CodeDeploy 代理程式。

> **Note**
>
> 您可以在主控台中建立部署群組時，設定 CodeDeploy 代理程式的自動安裝和更新。

啟動亞 Amazon EC2 執行個體 (CLI)

必要條件

如果您尚未這樣做，請按照中的說明設開始使用 CodeDeploy定和設定 AWS CLI 和建立 IAM 執行個體設定檔。

啟動 Amazon EC2 執行個體

1. 僅適用於 Windows 伺服器：如果您要建立執行 Windows 伺服器的 Amazon EC2 執行個體，請呼叫create-security-group和authorize-security-group-ingress命令以建立允許 RDP 存取的安全群組 (預設不允許存取)，或者也可以使用 HTTP 存取。例如，若要建立名為 CodeDeployDemo-Windows-Security 群組的安全性群組，請執行下列命令，一次執行一個命令：

   ```
   aws ec2 create-security-group --group-name CodeDeployDemo-Windows-Security-Group --
   description "For launching Windows Server images for use with CodeDeploy"
   ```

   ```
   aws ec2 authorize-security-group-ingress --group-name CodeDeployDemo-Windows-
   Security-Group --to-port 3389 --ip-protocol tcp --cidr-ip 0.0.0.0/0 --from-port
    3389
   ```

   ```
   aws ec2 authorize-security-group-ingress --group-name CodeDeployDemo-Windows-
   Security-Group --to-port 80 --ip-protocol tcp --cidr-ip 0.0.0.0/0 --from-port 80
   ```

   > **Note**
   >
   > 基於示範，這些命令會建立安全群組，以允許透過連接埠 3389 無限制地存取 RDP，或透過連接埠 80 無限制地存取 HTTP。最佳作法是，建議您限制對 RDP 和 HTTP 連接埠的存取。 CodeDeploy 不需要不受限制的連接埠存取，也不需要 HTTP 存取。如需詳細資訊，請參閱保護 Amazon EC2 執行個體的秘訣。

2. 呼叫命run-instances令以建立和啟動 Amazon EC2 執行個體。

   在您呼叫此命令之前，需要收集下列資訊：

- 您用於執行個體的 Amazon Machine Image (AMI) ID (*ami-id*)。若要取得 ID，請參閱尋找合適的 AMI。
- 您建立的 Amazon EC2 執行個體類型 (*######*) 的名稱，例如。`t1.micro`如需清單，請參閱 Amazon EC2 執行個體類型。
- 具有存取區域 CodeDeploy 代理程式安裝檔案之 Amazon S3 儲存貯體權限的 IAM 執行個體設定檔名稱。

  如需建立 IAM 執行個體設定檔的詳細資訊，請參閱步驟 4：為您的 Amazon EC2 執行個體建立 IAM 執行個體設定檔。
- Amazon EC2 執行個體 key pair (金*###) ###*，可讓您透過 SSH 存取執行 Amazon EC2 馬遜 Linux、Ubuntu 伺服器、RHEL 或 RDP 存取執行個體執行 Windows 伺服器的 Amazon EC2 執行個體。

  > **Important**
  >
  > 只輸入金鑰對名稱，而非金鑰對副檔名。例如，my-keypair，而非 my-keypair.pem。

  若要尋找 key pair 名稱，請在 https://console.aws.amazon.com/ec2 開啟 Amazon EC2 主控台。在導覽窗格中，於 Network & Security (網路與安全) 下選擇 Key Pairs (金鑰對)，然後記下清單中的金鑰對。

  若要產生 key pair，請參閱使用 Amazon EC2 建立 key pair。請務必在區域和中的端點中列出的其中一個區域中建立 key pair AWS 一般參考。否則，您將無法使用 Amazon EC2 執行個體 key pair CodeDeploy。

適用於 Amazon Linux、RHEL 和 Ubuntu 伺服器

呼叫run-instances命令以啟動執行 Amazon Linux、Ubuntu 伺服器或 RHEL 的亞馬遜 EC2 執行個體，並附加您在其中步驟 4：為您的 Amazon EC2 執行個體建立 IAM 執行個體設定檔建立的 IAM 執行個體設定檔。例如：

```
aws ec2 run-instances \
  --image-id ami-id \
  --key-name key-name \
  --count 1 \
  --instance-type instance-type \
```

```
  --iam-instance-profile Name=iam-instance-profile
```

> **Note**
>
> 此命令會為 Amazon EC2 執行個體建立一個預設安全群組，允許存取多個連接埠，包括透過連接埠 22 進行 SSH 的不受限制存取，也可以透過連接埠 80 進行 HTTP 存取。最佳作法是，我們建議您限制只能存取 SSH 和 HTTP 連接埠。 CodeDeploy 不需要不受限制的連接埠存取，也不需要 HTTP 連接埠存取。如需詳細資訊，請參閱保護 Amazon EC2 執行個體的秘訣。

視窗伺服器

呼叫run-instances命令以啟動執行 Windows Server 的 Amazon EC2 執行個體，並附加您在其中建立的 IAM 執行個體設定檔步驟 4：為您的 Amazon EC2 執行個體建立 IAM 執行個體設定檔，並指定您在步驟 1 中建立的安全群組名稱。例如：

```
aws ec2 run-instances --image-id ami-id --key-name key-name --count 1 --instance-
type instance-type --iam-instance-profile Name=iam-instance-profile --security-
groups CodeDeploy-Windows-Security-Group
```

這些命令會使用指定的 IAM 執行個體設定檔啟動具有指定 AMI、key pair 和執行個體類型的單一 Amazon EC2 執行個體，並在啟動期間執行指定的指令碼。

3. 請記下輸出中的 `InstanceID` 值。如果忘記此值，稍後可以透過針對 Amazon EC2 執行個體 key pair 呼叫describe-instances命令來取得此值。

```
aws ec2 describe-instances --filters "Name=key-name,Values=keyName" --query
 "Reservations[*].Instances[*].[InstanceId]" --output text
```

使用執行個體 ID 呼叫create-tags命令，該命令會標記 Amazon EC2 執行個體， CodeDeploy 以便稍後在部署期間找到它。在下列範例中，標籤已命名**`CodeDeployDemo`**，但您可以指定所需的任何 Amazon EC2 執行個體標籤。

```
aws ec2 create-tags --resources instance-id --tags Key=Name,Value=CodeDeployDemo
```

您可以同時將多個標籤套用至執行個體。例如：

```
aws ec2 create-tags --resources instance-id --tags Key=Name,Value=testInstance
 Key=Region,Value=West Key=Environment,Value=Beta
```

若要驗證 Amazon EC2 執行個體已啟動並通過所有檢查，請使用執行個體 ID 呼叫命describe-instance-status令。

```
aws ec2 describe-instance-status --instance-ids instance-id --query
 "InstanceStatuses[*].InstanceStatus.[Status]" --output text
```

如果執行個體已啟動並通過所有檢查，則 ok 會出現在輸出中。

安裝 CodeDeploy 代理程式

在 CodeDeploy部署中使用 CodeDeploy 代理程式之前，必須先在 Amazon EC2 執行個體上安裝代理程式。如需詳細資訊，請參閱 安裝 CodeDeploy 代理程式。

> **Note**
>
> 您可以在主控台中建立部署群組時，設定 CodeDeploy 代理程式的自動安裝和更新。

為 CodeDeploy (AWS CloudFormation 範本) 建立 Amazon EC2 執行個體

您可以使用我們的 AWS CloudFormation 範本快速啟動執行 Amazon Linux 或視窗伺服器的 Amazon EC2 執行個體。您可以使用 AWS CLI、主 CodeDeploy 控台或 AWS API 透過範本啟動執行個體。除了啟動執行個體之外，範本還會執行下列動作：

- 指示授 AWS CloudFormation 與執行個體參與 CodeDeploy部署的權限。
- 標記執行個體， CodeDeploy 以便在部署期間找到它。
- 在執行個體上安裝並執行 CodeDeploy 代理程式。

您不必使用我們的設 AWS CloudFormation 定 Amazon EC2 執行個體。如需替代方案，請參閱使用的例證 CodeDeploy。

我們不會為執行 Ubuntu 伺服器或 RHEL (RHEL) 的 Amazon EC2 執行個體提供 AWS CloudFormation 範本。

主題

開始之前

在您可以使用 AWS CloudFormation 範本啟動 Amazon EC2 執行個體之前，請務必完成以下步驟。

1. 請確定您已建立管理使用者，如中所述步驟 1：設定。仔細檢查用戶是否具有以下最低權限，並添加任何不存在的權限：

   - 雲形：*
   - codedeploy:*
   - ec2:*
   - IAM：AddRoleToInstanceProfile
   - IAM：CreateInstanceProfile
   - IAM：CreateRole
   - IAM：DeleteInstanceProfile
   - IAM：DeleteRole
   - IAM：DeleteRolePolicy
   - IAM：GetRole
   - IAM：DeleteRolePolicy
   - IAM：PutRolePolicy
   - IAM：RemoveRoleFromInstanceProfile
2. 請確定您擁有執行個體金鑰組，以啟用對執行 Amazon Linux 之 Amazon EC2 執行個體的 SSH 存取權，或對執行 Windows 伺服器執行個體的 RDP 存取權。

   若要尋找 key pair 名稱，請在 https://console.aws.amazon.com/ec2 開啟 Amazon EC2 主控台。在導覽窗格中，於 Network & Security (網路與安全) 下選擇 Key Pairs (金鑰對)，然後記下清單中的金鑰對。

   若要產生新的 key pair，請參閱使用 Amazon EC2 建立 key pair。請確定 key pair 是在區域和中的端點中列出的其中一個區域中建立AWS 一般參考。否則，您無法搭配使用執行個體 key pair CodeDeploy。

使用 AWS CloudFormation 範本 (主控台) 啟動 Amazon EC2 執行個體

1. 請登入 AWS Management Console 並開啟 AWS CloudFormation 主控台，網址為 https://console.aws.amazon.com/cloudformation。

   > **Important**
   >
   > 使用您所使 AWS Management Console 用的相同帳戶登入開始使用 CodeDeploy。在導覽列的區域選取器中，選擇 [區域] 中列出的其中一個區域和中的端點AWS 一般參考。CodeDeploy 僅支援這些區域。

2. 選擇 Create Stack (建立堆疊)。
3. 在 [選擇範本] 中，選擇 [指定 Amazon S3 範本網址]。在方塊中，輸入您所在地區的 AWS CloudFormation 範本位置，然後選擇 [下一步]。

區域	AWS CloudFormation 範本的位置
美國東部 (俄亥俄) 區域	`http://s3-us-east-2.amazonaws.com/aws-codedeploy-us-east-2/templates/latest/CodeDeploy_SampleCF_Template.json`
美國東部 (維吉尼亞北部) 區域	`http://s3.amazonaws.com/aws-codedeploy-us-east-1/templates/latest/CodeDeploy_SampleCF_Template.json`
美國西部 (加利佛尼亞北部) 區域	`http://s3-us-west-1.amazonaws.com/aws-codedeploy-us-west-1/templates/latest/CodeDeploy_SampleCF_Template.json`
美國西部 (奧勒岡) 區域	`http://s3-us-west-2.amazonaws.com/aws-codedeploy-us-west-2/templates/latest/CodeDeploy_SampleCF_Template.json`

區域	AWS CloudFormation 範本的位置
加拿大 (中部) 區域	http://s3-ca-central-1.amazonaws.com/aws-codedeploy-ca-central-1/templates/latest/CodeDeploy_SampleCF_Template.json
歐洲 (愛爾蘭) 區域	http://s3-eu-west-1.amazonaws.com/aws-codedeploy-eu-west-1/templates/latest/CodeDeploy_SampleCF_Template.json
歐洲 (倫敦) 區域	http://s3-eu-west-2.amazonaws.com/aws-codedeploy-eu-west-2/templates/latest/CodeDeploy_SampleCF_Template.json
歐洲 (巴黎) 區域	http://s3-eu-west-3.amazonaws.com/aws-codedeploy-eu-west-3/templates/latest/CodeDeploy_SampleCF_Template.json
歐洲 (法蘭克福) 區域	http://s3-eu-central-1.amazonaws.com/aws-codedeploy-eu-central-1/templates/latest/CodeDeploy_SampleCF_Template.json
以色列 (特拉維夫) 區域	http://s3-il-central-1.amazonaws.com/aws-codedeploy-il-central-1/templates/latest/CodeDeploy_SampleCF_Template.json

區域	AWS CloudFormation 範本的位置
亞太區域 (香港) 區域	http://s3-ap-east-1.amazonaws.com/aws-codedeploy-ap-east-1/templates/latest/CodeDeploy_SampleCF_Template.json
亞太區域 (東京) 區域	http://s3-ap-northeast-1.amazonaws.com/aws-codedeploy-ap-northeast-1/templates/latest/CodeDeploy_SampleCF_Template.json
亞太區域 (首爾) 區域	http://s3-ap-northeast-2.amazonaws.com/aws-codedeploy-ap-northeast-2/templates/latest/CodeDeploy_SampleCF_Template.json
亞太區域 (新加坡) 區域	http://s3-ap-southeast-1.amazonaws.com/aws-codedeploy-ap-southeast-1/templates/latest/CodeDeploy_SampleCF_Template.json
亞太區域 (雪梨) 區域	http://s3-ap-southeast-2.amazonaws.com/aws-codedeploy-ap-southeast-2/templates/latest/CodeDeploy_SampleCF_Template.json
亞太區域 (墨爾本) 區域	https://aws-codedeploy-ap-southeast-4.s3.ap-southeast-4.amazonaws.com/templates/latest/CodeDeploy_SampleCF_Template.json

區域	AWS CloudFormation 範本的位置
亞太 (孟買) 區域	`http://s3-ap-south-1.amazonaws.com/aws-codedeploy-ap-south-1/templates/latest/CodeDeploy_SampleCF_Template.json`
南美洲 (聖保羅) 區域	`aws-codedeploy-ap-northeast-1.s3.sa-east-1.amazonaws.com/templates/latest/CodeDeploy_SampleCF_Template.json`

4. 在 Stack name (堆疊名稱) 方塊中，輸入堆疊的名稱 (例如，**`CodeDeployDemoStack`**)。
5. 在 Parameters (參數) 中，輸入下列資訊，然後選擇 Next (下一步)。
   - 針對 InstanceCount，輸入您要啟動的執行個體數目。(建議您保留預設值 1)。
   - 針對 InstanceType，輸入您要啟動的執行個體類型 (或保留預設值 t1.micro)。
   - 在中 KeyPairName，輸入執行個體 key pair 名稱。只輸入金鑰對名稱，而非金鑰對副檔名。
   - 對於 OperatingSystem「方塊」，請鍵入**`Windows`**以啟動執行 Windows 伺服器的執行個體 (或保留 Linux 的預設值)。
   - 針對 SSHLocation，輸入 IP 地址範圍，以用於使用 SSH 或 RDP 連線至執行個體 (或保留預設值 0.0.0.0/0)。

     > **Important**
     >
     > 提供的**`0.0.0.0/0`**預設值僅用於展示目的。 CodeDeploy 不需要 Amazon EC2 執行個體不受限制地存取連接埠。根據最佳實務，建議您限制對 SSH (和 HTTP) 連接埠的存取。如需詳細資訊，請參閱保護 Amazon EC2 執行個體的秘訣。

   - 對於 TagKey，輸入部署期間 CodeDeploy 要用來識別執行個體的執行個體標記金鑰 (或保留「名稱」的預設值)。
   - 針對 TagValue，輸入在部署期間用 CodeDeploy 來識別執行個體的執行個體標記值 (或保留預設值 CodeDeployDemo)。
6. 在 Options (選項) 頁面上，將選項方塊空白，然後選擇 Next (下一步)。

> **Important**
>
> AWS CloudFormation 標籤與標 CodeDeploy 籤不同。 AWS CloudFormation 使用標籤來簡化基礎結構的管理。 CodeDeploy 使用標籤來識別 Amazon EC2 執行個體。您在「指定參數」頁面上指定了 CodeDeploy標籤。

7. 在 [檢閱] 頁面的 [功能] 中，選取 [我確認 AWS CloudFormation 可能會建立 IAM 資源] 方塊，然後選擇 [建立]。

   建立堆疊並啟動 Amazon EC2 執行個體之後 AWS CloudFormation ，在 AWS CloudFormation 主控台中，「狀態」欄中會顯示 CREATE_COMPLETE。此程序需要幾分鐘的時間。

若要確認 CodeDeploy 代理程式是否在 Amazon EC2 執行個體上執行，請參閱管理 CodeDeploy 代理程式作，然後繼續執行建立應用程式 CodeDeploy。

使用 AWS CloudFormation 範本 (AWS CLI) 啟動 Amazon EC2 執行個體

1. 在呼叫create-stack命令時使用我們的 AWS CloudFormation 範本。此堆疊將啟動已安裝 CodeDeploy 代理程式的新 Amazon EC2 執行個體。

   若要啟動執行 Amazon Linux 的亞馬遜 EC2 執行個體：

```
aws cloudformation create-stack \
  --stack-name CodeDeployDemoStack \
  --template-url templateURL \
  --parameters ParameterKey=InstanceCount,ParameterValue=1
 ParameterKey=InstanceType,ParameterValue=t1.micro \
    ParameterKey=KeyPairName,ParameterValue=keyName
 ParameterKey=OperatingSystem,ParameterValue=Linux \
    ParameterKey=SSHLocation,ParameterValue=0.0.0.0/0
 ParameterKey=TagKey,ParameterValue=Name \
    ParameterKey=TagValue,ParameterValue=CodeDeployDemo \
  --capabilities CAPABILITY_IAM
```

   若要啟動執行視窗伺服器的 Amazon EC2 執行個體：

```
aws cloudformation create-stack --stack-name CodeDeployDemoStack --template-
url template-url --parameters ParameterKey=InstanceCount,ParameterValue=1
 ParameterKey=InstanceType,ParameterValue=t1.micro
```

```
ParameterKey=KeyPairName,ParameterValue=keyName
ParameterKey=OperatingSystem,ParameterValue=Windows
ParameterKey=SSHLocation,ParameterValue=0.0.0.0/0
ParameterKey=TagKey,ParameterValue=Name
ParameterKey=TagValue,ParameterValue=CodeDeployDemo --capabilities CAPABILITY_IAM
```

金###是執行個體 key pair 名稱。只輸入金鑰對名稱，而非金鑰對副檔名。

####是您所在地區的 AWS CloudFormation 模板的位置：

區域	AWS CloudFormation 範本的位置
美國東部 (俄亥俄) 區域	http://s3-us-east-2.amazonaws.com/aws-codedeploy-us-east-2/templates/latest/CodeDeploy_SampleCF_Template.json
美國東部 (維吉尼亞北部) 區域	http://s3.amazonaws.com/aws-codedeploy-us-east-1/templates/latest/CodeDeploy_SampleCF_Template.json
美國西部 (加利佛尼亞北部) 區域	http://s3-us-west-1.amazonaws.com/aws-codedeploy-us-west-1/templates/latest/CodeDeploy_SampleCF_Template.json
美國西部 (奧勒岡) 區域	http://s3-us-west-2.amazonaws.com/aws-codedeploy-us-west-2/templates/latest/CodeDeploy_SampleCF_Template.json
加拿大 (中部) 區域	http://s3-ca-central-1.amazonaws.com/aws-codedeploy-ca-central-1/templates/latest/CodeDeploy_SampleCF_Template.json

區域	AWS CloudFormation 範本的位置
歐洲 (愛爾蘭) 區域	http://s3-eu-west-1.amazonaws.com/aws-codedeploy-eu-west-1/templates/latest/CodeDeploy_SampleCF_Template.json
歐洲 (倫敦) 區域	http://s3-eu-west-2.amazonaws.com/aws-codedeploy-eu-west-2/templates/latest/CodeDeploy_SampleCF_Template.json
歐洲 (巴黎) 區域	http://s3-eu-west-3.amazonaws.com/aws-codedeploy-eu-west-3/templates/latest/CodeDeploy_SampleCF_Template.json
歐洲 (法蘭克福) 區域	http://s3-eu-central-1.amazonaws.com/aws-codedeploy-eu-central-1/templates/latest/CodeDeploy_SampleCF_Template.json
以色列 (特拉維夫) 區域	http://s3-il-central-1.amazonaws.com/aws-codedeploy-il-central-1/templates/latest/CodeDeploy_SampleCF_Template.json
亞太區域 (香港) 區域	http://s3-ap-east-1.amazonaws.com/aws-codedeploy-ap-east-1/templates/latest/CodeDeploy_SampleCF_Template.json

區域	AWS CloudFormation 範本的位置
亞太區域 (東京) 區域	http://s3-ap-northeast-1.amazonaws.com/aws-codedeploy-ap-northeast-1/templates/latest/CodeDeploy_SampleCF_Template.json
亞太區域 (首爾) 區域	http://s3-ap-northeast-2.amazonaws.com/aws-codedeploy-ap-northeast-2/templates/latest/CodeDeploy_SampleCF_Template.json
亞太區域 (新加坡) 區域	http://s3-ap-southeast-1.amazonaws.com/aws-codedeploy-ap-southeast-1/templates/latest/CodeDeploy_SampleCF_Template.json
亞太區域 (雪梨) 區域	http://s3-ap-southeast-2.amazonaws.com/aws-codedeploy-ap-southeast-2/templates/latest/CodeDeploy_SampleCF_Template.json
亞太區域 (墨爾本) 區域	https://aws-codedeploy-ap-southeast-4.s3.ap-southeast-4.amazonaws.com/templates/latest/CodeDeploy_SampleCF_Template.json
亞太 (孟買) 區域	http://s3-ap-south-1.amazonaws.com/aws-codedeploy-ap-south-1/templates/latest/CodeDeploy_SampleCF_Template.json

區域	AWS CloudFormation 範本的位置
南美洲 (聖保羅) 區域	`aws-codedeploy-ap-northeast-1.s3.sa-east-1.amazonaws.com/templates/latest/CodeDeploy_SampleCF_Template.json`

此命令會使用指定 Amazon S3 儲存貯體中的 AWS CloudFormation 範本建立名**CodeDeployDemoStack**為的 AWS CloudFormation 堆疊。Amazon EC2 執行個體以 t1.micro 執行個體類型為基礎，但您可以使用任何類型。它會加上 **CodeDeployDemo** 值的標籤，但您可以將它加上任何值的標籤。它已套用指定的執行個體金鑰對。

2. 調用命describe-stacks令以驗證命名的 AWS CloudFormation 堆棧**CodeDeployDemoStack**是否成功創建：

```
aws cloudformation describe-stacks --stack-name CodeDeployDemoStack --query
 "Stacks[0].StackStatus" --output text
```

傳回 `CREATE_COMPLETE` 值之前，請不要繼續進行。

若要確認 CodeDeploy 代理程式是否在 Amazon EC2 執行個體上執行，請參閱管理 CodeDeploy 代理程式作，然後繼續執行建立應用程式 CodeDeploy。

設定要使用的 Amazon EC2 執行個體 CodeDeploy

這些指示說明如何設定執行 Amazon EC2 執行個體，執行 Amazon Linux、Ubuntu 伺服器、RHEL 或視窗伺服器，以便在 CodeDeploy 部署中使用。

> **Note**
>
> 如果您沒有 Amazon EC2 實例，則可以使用該 AWS CloudFormation 模板啟動一個運行 Amazon Linux 或 Windows 服務器的實例。我們不提供 Ubuntu 服務器或 RHEL 的模板。

步驟 1：確認 IAM 執行個體設定檔已連接至您的 Amazon EC2 執行個體

1. 登入 AWS Management Console 並開啟 Amazon EC2 主控台，網址為 https://console.aws.amazon.com/ec2/。

2. 在導覽窗格的 Instances (執行個體) 下方，選擇 Instances (執行個體)。
3. 在清單中瀏覽並選擇您的 Amazon EC2 執行個體。
4. 在詳細資料窗格的 [說明] 索引標籤上，記下 IAM 角色欄位中的值，然後繼續下一節。

   如果欄位為空白，您可以將 IAM 執行個體設定檔附加到執行個體。如需詳細資訊，請參閱將 IAM 角色附加至執行個體。

步驟 2：確認附加的 IAM 執行個體設定檔具有正確的存取權限

1. 在以下網址開啟 IAM 主控台：https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇角色。
3. 瀏覽並選擇您在上一節步驟 4 中記下的 IAM 角色名稱。

> **Note**
>
> 如果您要使用 AWS CloudFormation 範本所產生的服務角色，而不是依照中的指示建立的服務角色步驟 2：建立服務角色 CodeDeploy，請注意下列事項：
> 在某些版本的 AWS CloudFormation 範本中，產生並連接到 Amazon EC2 執行個體的 IAM 執行個體設定檔的顯示名稱與 IAM 主控台中的顯示名稱不同。例如，IAM 執行個體設定檔的顯示名稱可能為`CodeDeploySampleStack-expnyi6-InstanceRoleInstanceProfile-IK8J8A9123EX`，而 IAM 主控台中的 IAM 執行個體設定檔的顯示名稱可能為`CodeDeploySampleStack-expnyi6-InstanceRole-C5P33V1L64EX`。
> 為了協助您識別 IAM 主控台中的執行個體設定檔，您會看到兩者`CodeDeploySampleStack-expnyi6-InstanceRole`的前置字元相同。如需這些顯示名稱為何不同的詳細資訊，請參閱執行個體設定檔。

4. 選取 Trust Relationships (信任關係) 索引標籤。如果受信任的實體中沒有讀取身分識別提供者 ec2.amazonaws.com 的項目，則無法使用此 Amazon EC2 執行個體。使用中的資訊停止並建立 Amazon EC2 執行個體使用的例證 CodeDeploy。

   如果有一個讀取身份提供程序 ec2.amazonaws.com 的條目，並且您僅將應用程序存儲在存儲 GitHub庫中，然後跳到。步驟 3：標記 Amazon EC2 實例

   如果有一個讀取身分識別提供者 ec2.amazonaws.com 的項目，而您要將應用程式存放在 Amazon S3 儲存貯體中，請選擇「權限」索引標籤。

5. 如果 Managed Policies (受管政策) 區域中有政策，則請展開政策，然後選擇 Edit policy (編輯政策)。
6. 選擇 JSON 標籤。如果您要將應用程式存放在 Amazon S3 儲存貯體中，請確定`"s3:Get*"`並列`"s3:List*"`在指定動作清單中。

   這可能看起來如下：

   ```
   {"Statement":[{"Resource":"*","Action":[
     ... Some actions may already be listed here ...
     "s3:Get*","s3:List*"
     ... Some more actions may already be listed here ...
     ],"Effect":"Allow"}]}
   ```

   或者，這可能看起來如下：

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
         {
           "Action": [
             ... Some actions may already be listed here ...
             "s3:Get*",
             "s3:List*"
             ... Some more actions may already be listed here ...
           ],
           ...
         }
       ]
     }
   ```

   如果 `"s3:Get*"` 和 `"s3:List*"` 不在指定的動作清單中，請選擇 Edit (編輯) 新增它們，然後選擇 Save (儲存) (如果 `"s3:Get*"` 和 `"s3:List*"` 都不是清單中的最後一個動作，則請務必在動作後面新增逗號，才能驗證政策文件)。

   > **Note**
   >
   > 我們建議您將此政策限制為只有 Amazon EC2 執行個體必須存取的 Amazon S3 儲存貯體。確保授予對包含 CodeDeploy 代理程式的 Amazon S3 儲存貯體的存取權。否則，在執行個體上安裝或更新 CodeDeploy 代理程式時，可能會發生錯誤。若要僅授與 IAM 執行

個體設定檔存取權限給 Amazon S3 中某些 CodeDeploy 資源套件儲存貯體，請使用下列政策，但移除要阻止存取的儲存貯體的行：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3:::replace-with-your-s3-bucket-name/*",
        "arn:aws:s3:::aws-codedeploy-us-east-2/*",
        "arn:aws:s3:::aws-codedeploy-us-east-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-2/*",
        "arn:aws:s3:::aws-codedeploy-ca-central-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-3/*",
        "arn:aws:s3:::aws-codedeploy-eu-central-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-central-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-north-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-south-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-south-2/*",
        "arn:aws:s3:::aws-codedeploy-il-central-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-east-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-3/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-3/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-4/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-2/*",
        "arn:aws:s3:::aws-codedeploy-me-central-1/*",
        "arn:aws:s3:::aws-codedeploy-me-south-1/*",
        "arn:aws:s3:::aws-codedeploy-sa-east-1/*"
      ]
    }
```

```
    ]
}
```

步驟 3：標記 Amazon EC2 實例

如需如何標記 Amazon EC2 執行個體 CodeDeploy 以便在部署期間找到它的指示，請參閱在主控台中使用標籤，然後返回此頁面。

> **Note**
>
> 您可以使用您喜歡的任何金鑰和值來標記 Amazon EC2 執行個體。只務必在您部署至其中時，指定此金鑰和值。

步驟 4：在亞馬遜 EC2 執行個體上安裝 AWS CodeDeploy 代理程式

如需如何在 Amazon EC2 執行個體上安裝 CodeDeploy 代理程式並驗證代理程式是否正在執行的指示，請參閱管理 CodeDeploy 代理程式作，然後繼續執行建立應用程式 CodeDeploy。

使用內部部署執行個體 CodeDeploy

現場部署執行個體是任何非 Amazon EC2 執行個體的實體裝置，可以執行 CodeDeploy 代理程式並連接到公有 AWS 服務端點。

將 CodeDeploy 應用程式修訂部署到內部部署執行個體包含兩個主要步驟：

- 步驟 1 — 設定每個內部部署執行個體、向其註冊 CodeDeploy，然後加上標記。
- 步驟 2 — 將應用程式修訂部署到內部部署執行個體。

> **Note**
>
> 若要試驗建立和部署範例應用程式修訂版，以正確設定和註冊現場部署執行個體，請參閱教學課程：使用 CodeDeploy (Windows 伺服器、Ubuntu 伺服器或 RHEL) 將應用程式部署到內部部署執行個體。如需內部部署執行個體及其使用方式的相關資訊 CodeDeploy，請參閱Working with On-Premises Instances。

如果您不想在部署中再使用內部部署執行個體，可以從部署群組中移除內部部署執行個體標記。如需更強力的方法，請從執行個體移除現場部署執行個體標籤。您也可以明確撤銷現場部署執行個體的註冊，如此，則該現場部署執行個體不能再嫆於任何部署。如需詳細資訊，請參閱 管理內部部署執行個體作 CodeDeploy。

本節中的指示說明如何設定內部部署執行個體，然後註冊並加上標記，以 CodeDeploy 便在部署中使用。本節也說明如何在您不再打算部署到內部部署執行個體之後取得內部部署執行個體的相關資訊，以及如何取消註冊內部部署執行個體。 CodeDeploy

主題

設定內部部署執行個體的前提

在您可以註冊現場部署執行個體前，必須滿足以下先決條件。

> **Important**
>
> 如果您使用register-on-premises-instance命令並定期重新整理使用 AWS Security Token Service (AWS STS) 產生的臨時認證，還有其他先決條件。如需相關資訊，請參閱IAM 工作階段 ARN 註冊先決條件。

裝置要求

您要準備、註冊和標記為內部部署執行個體的裝置 CodeDeploy 必須執行支援的作業系統。如需清單，請參閱 CodeDeploy 代理程式支援的作業系統。

如果您的作業系統不受支援， CodeDeploy 代理程式會以開放原始碼的形式提供，以便您適應您的需求。如需詳細資訊，請參閱中的CodeDeploy 代理程式儲存庫 GitHub。

對外通訊

內部部署執行個體必須能夠連線到公用 AWS 服務端點，才能與之通訊 CodeDeploy。

CodeDeploy 代理程式透過連接埠 443 使用 HTTPS 進行輸出通訊。

管理控制

在內部部署執行個體上用來設定內部部署執行個體的本機或網路帳戶，必須能夠以sudo或身分執行root (針對 Ubuntu Server) 或以系統管理員身分執行 (適用於 Windows Server)。

IAM 許可

您用來註冊現場部署執行個體的 IAM 身分必須獲得完成註冊的權限 (並視需要取消註冊內部部署執行個體)。

除了中所述的政策之外步驟 3：限制 CodeDeploy 使用者的權限，請確定呼叫的 IAM 身分已附加下列其他政策。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateAccessKey",
        "iam:CreateUser",
        "iam:DeleteAccessKey",
        "iam:DeleteUser",
        "iam:DeleteUserPolicy",
        "iam:ListAccessKeys",
        "iam:ListUserPolicies",
        "iam:PutUserPolicy",
        "iam:GetUser"
      ],
      "Resource": "*"
    }
  ]
}
```

如需關於如何附加 IAM 政策的資訊，請參閱管理 IAM 政策。

註冊內部部署執行個體 CodeDeploy

若要註冊現場部署執行個體，您必須使用 IAM 身分來驗證您的請求。您可以從以下選項中選擇使用的 IAM 身分和註冊方式：

- 使用 IAM 角色 ARN 來驗證請求。
  - 使用命register-on-premises-instance令並定期重新整理使用 AWS Security Token Service (AWS STS) 產生的臨時認證，以手動設定大部分的註冊選項。此選項提供最高級別的安全性，因為使用

臨時令牌進行身份驗證，該權杖會逾時且必須定期重新整理。對於任何規模的生產部署，建議使用此選項。如需相關資訊，請參閱使用命 register-on-premises-instance 令 (IAM 工作階段 ARN) 註冊內部部署執行個體。

- (不建議) 使用 IAM 使用者 ARN 來驗證請求。
  - 使用 register 命令進行最自動化的註冊過程。此選項僅適用於不太考量安全性的非生產部署。此選項較不安全，因為它使用靜態 (永久) 認證進行驗證。此選項適用於註冊單一內部部署執行個體。如需相關資訊，請參閱使用註冊命令 (IAM 使用者 ARN) 註冊內部部署執行個體。
  - 使用指register-on-premises-instance令手動設定大多數註冊選項。適用於註冊少量的現場部署執行個體。如需相關資訊，請參閱使用命 register-on-premises-instance 令 (IAM 使用者 ARN) 註冊內部部署執行個體。

主題

使用命 register-on-premises-instance 令 (IAM 工作階段 ARN) 註冊內部部署執行個體

為了最大限度地控制內部部署執行個體的驗證和註冊，您可以使用register-on-premises-instance命令並定期重新整理使用 AWS Security Token Service (AWS STS) 產生的臨時認證。執行個體的靜態 IAM 角色扮演這些重新整理 AWS STS 登入資料的角色，以執行 CodeDeploy 部署作業。

當您需要註冊大量執行個體時，此方法非常有用。它允許您使用自動化註冊過程 CodeDeploy。您可以使用自己的身分識別和身分驗證系統來驗證內部部署執行個體，並將 IAM 工作階段登入資料從服務分發到執行個體以搭配 CodeDeploy使用

> **Note**
>
> 或者，您可以使用分散至所有內部部署執行個體的共用 IAM 使用者呼叫 AWS STS AssumeRoleAPI 來擷取內部部署執行個體的工作階段登入資料。此方法安全性較低，不建議用於生產或任務關鍵環境。

您可以使用下列主題中的資訊，使用產生的臨時安全性認證來設定內部部署執行個體 AWS STS。

主題

IAM 工作階段 ARN 註冊先決條件

除了列於設定內部部署執行個體的前提中的必要條件之外，也必須符合下列其他要求：

IAM 許可

您用來註冊現場部署執行個體的 IAM 身分必須獲得執行 CodeDeploy 作業的許可。確定受AWSCodeDeployFullAccess管政策已附加至 IAM 身分。如需詳細資訊，請參閱 AWS IAM 使用者指南中的受管政策。

用於重新整理臨時登入資料的系統

如果您使用 IAM 工作階段 ARN 註冊現場部署執行個體，您必須擁有一個定期重新整理臨時登入資料的系統。如果在產生登入資料時指定的期間較短，臨時登入資料會在一小時 (或更短時間) 後過期。重新整理登入資料的方法有兩種：

- 方法 1：使用您公司網路中的身分和身分驗證系統以及 CRON 指令碼，該指令碼會定期輪詢該身分和身分驗證系統，並將最新的工作階段登入資料複製到該執行個體。這可讓您將驗證和身分結構與整合， AWS 而無需變更 CodeDeploy 代理程式或服務，以支援您在組織中使用的驗證類型。
- 方法 2：定期在執行個體上執行 CRON 工作以呼叫 AWS STS AssumeRole動作，並將工作階段認證寫入 CodeDeploy 代理程式可存取的檔案。此方法仍需要使用 IAM 使用者並將登入資料複製到現場部署執行個體，但您可以對各個現場部署執行個體機群重複使用相同的 IAM 使用者和登入資料。

> **Note**
>
> 無論您使用的是方法 1 還是 2，都必須設定處理程序，以便在臨時工作階段認證更新後重新啟動 CodeDeploy 代理程式，以便新認證生效。

如需建立和使用 AWS STS 認證的相關資訊，請參閱 AWS Security Token Service API 參考和使用臨時安全登入資料來要求 AWS 資源存取權。

步驟 1：建立內部部署執行個體將承擔的 IAM 角色

您可以使用 AWS CLI 或 IAM 主控台建立 IAM 角色，讓內部部署執行個體使用該角色進行驗證和互動 CodeDeploy。

您只需建立一個 IAM 角色。您的每個現場部署執行個體都可以擔任此角色，以擷取為此角色提供許可的臨時安全登入資料。

您建立的角色需要下列權限，才能存取安裝 CodeDeploy 代理程式所需的檔案：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

建議您將此政策限制為只有現場部署執行個體需要存取的 Amazon S3 儲存貯體。如果限制此政策，請確保授予對包含 CodeDeploy 代理程式之 Amazon S3 儲存貯體的存取權。否則，每當在內部部署執行個體上安裝或更新 CodeDeploy 代理程式時，就可能會發生錯誤。如需控制 Amazon S3 儲存貯體存取的相關資訊，請參閱管理 Amazon S3 資源的存取許可。

建立 IAM 角色

1. 使用--role-name選項來指定 IAM 角色名稱 (例如CodeDeployInstanceRole) 和提供權限的--assume-role-policy-document選項，呼叫 create- role 命令。

   當您為此執行個體建立 IAM 角色時，您可以將其命名為 CodeDeployInstanceRole，並在名為 CodeDeployRolePolicy.json 的檔案中提供所需許可：

   ```
   aws iam create-role --role-name CodeDeployInstanceRole --assume-role-policy-
   document file://CodeDeployRolePolicy.json
   ```

2. 在呼叫 create-role 命令的輸出中，記錄 ARN 欄位的值。例如：

   ```
   arn:aws:iam::123456789012:role/CodeDeployInstanceRole
   ```

   當您使用 AWS STS AssumeRoleAPI 產生每個執行個體的短期認證時，您將需要角色 ARN。

   如需有關建立 IAM 角色的詳細資訊，請參閱《IAM 使用者指南》中的建立角色以將許可委派給 AWS 服務。

   如需將權限指派給現有角色的資訊，請參閱《AWS CLI 命令參考》put-role-policy中的〈〉。

步驟 2：使用以下方式為個別執行個體產生臨時認證 AWS STS

在您產生將用於註冊現場部署執行個體的臨時登入資料之前，必須先建立或選擇您將為其產生臨時登入資料的 IAM 身分 (使用者或角色)。在此 IAM 身分的政策設定中，必須包含 sts:AssumeRole 許可。

如需授與 IAM 身分sts:AssumeRole權限的相關資訊，請參閱建立角色以將權限委派給 AWS 服務和AssumeRole。

有兩種方式可產生臨時登入資料：

- 搭配使用假設角色指令。 AWS CLI例如：

  ```
  aws sts assume-role --role-arn arn:aws:iam::12345ACCOUNT:role/role-arn --role-
  session-name session-name
  ```

  其中：

  - *12345ACCOUNT* 是您組織的 12 位數號碼。
  - *role-arn* 是要擔任的角色 ARN (在步驟 1：建立內部部署執行個體將承擔的 IAM 角色中產生)。

- *session-name* 是您為正在建立之角色工作階段提供的名稱。

> **Note**
>
> 如果您使用定期輪詢身分識別和驗證系統的 CRON 指令碼，並將最新的工作階段認證複製到執行個體 (方法 1 用於重新整理臨時認證，請改為使用任何支援的 AWS SDK 來呼叫IAM 工作階段 ARN 註冊先決條件)。AssumeRole

- 使用提供的工具 AWS。

  此工 aws-codedeploy-session-helper 具會產生 AWS STS 認證，並將其寫入您放置在執行個體上的檔案。此工具最適用於 IAM 工作階段 ARN 註冊先決條件中所述用於重新整理臨時登入資料的方法 2。在此方法中， aws-codedeploy-session-helper 工具會放置在每個執行個體上，並使用 IAM 使用者的許可執行命令。每個執行個體都使用與此工具相同的 IAM 使用者登入資料。

  如需詳細資訊，請參閱存aws-codedeploy-session-helper GitHub 放庫。

> **Note**
>
> 在您建立 IAM 工作階段登入資料後，將其放置在現場部署執行個體上的任何位置。在下一個步驟中，您將設定 CodeDeploy 代理程式以存取此位置中的認證。

在繼續之前，請確保您將用於定期重新整理臨時登入資料的系統已準備好。如果臨時登入資料未重新整理，對現場部署執行個體的部署將會失敗。如需詳細資訊，請參閱 IAM 工作階段 ARN 註冊先決條件中「用於重新整理臨時登入資料的系統」。

步驟 3：將設定檔新增至內部部署執行個體

使用 root 或管理員許可，將組態檔案新增至現場部署執行個體。此設定檔用於宣告 IAM 登入資料和要用於的目標 AWS 區域 CodeDeploy。該檔案必須新增至現場部署執行個體上的特定位置。檔案必須包含 IAM 臨時工作階段 ARN、其秘密金鑰 ID 和秘密存取金鑰，以及目標 AWS 區域。

新增組態檔案

1. 在內部部署執行個體的下列位置建立名為 `codedeploy.onpremises.yml` (針對 Ubuntu 伺服器或 RHEL 內部部署執行個體) 或 `conf.onpremises.yml` (適用於 Windows Server 內部部署執行個體) 的檔案：

   - 對於 Ubuntu 服務器：`/etc/codedeploy-agent/conf`

- 對於視窗伺服器：C:\ProgramData\Amazon\CodeDeploy

2. 使用文字編輯器將下列資訊新增至新建立的codedeploy.onpremises.yml檔案 (Linux) 或conf.onpremises.yml檔案 (Windows)：

```
---
iam_session_arn: iam-session-arn
aws_credentials_file: credentials-file
region: supported-region
```

其中：

- *iam-session-arn*是您在中步驟 2：使用以下方式為個別執行個體產生臨時認證 AWS STS註明的 IAM 工作階段 ARN。
- *credentials-file* 是在步驟 2：使用以下方式為個別執行個體產生臨時認證 AWS STS中記錄的臨時工作階段 ARN 登入資料檔案所在位置。
- 支*###*是 CodeDeploy支援的區域之一，如中的區域和端點中所列。AWS 一般參考

步驟 4：準備用於部署的內部部 CodeDeploy 署執行個體

安裝和配置 AWS CLI

在內部部署執行個體 AWS CLI 上安裝和設定。(AWS CLI 將用於在內部部署執行個體上下載和安裝 CodeDeploy 代理程式。)

1. 若要在內部部署執行個體 AWS CLI 上安裝，請依照AWS Command Line Interface 使用者指南中的[取得設定] AWS CLI 中的指示進行。

> **Note**
>
> CodeDeploy 用於使用內部部署執行個體的命令在. AWS CLI如果您已 AWS CLI 經安裝了版本，則可以通過調用來檢查其版本aws --version。

2. 若要在內部部署執行個體 AWS CLI 上設定，請依照AWS Command Line Interface 使用者指南中的〈設定〉 AWS CLI中的指示進行。

> **Important**
>
> 在設定 AWS CLI (例如，透過呼叫aws configure命令) 時，請務必指定 IAM 使用者的秘密金鑰 ID 和秘密存取金鑰，這些使用者至少具有中所述的權限IAM 工作階段 ARN 註冊先決條件。

設定 AWS_REGION 環境變數 (僅適用於 Ubuntu Server 和 RHEL)

如果您沒有在內部部署執行個體上執行 Ubuntu Server 或 RHEL，請略過此步驟並直接前往「安裝 CodeDeploy 代理程式」。

在 Ubuntu 伺服器或 RHEL 內部部署執行個體上安裝 CodeDeploy 代理程式，並讓執行個體在有新版本可用時更新 CodeDeploy 代理程式。您可以透過將執行個體上的AWS_REGION環境變數設定為其中一個支援區域的識別碼來執行此操作 CodeDeploy。建議您將值設定為 CodeDeploy 應用程式、部署群組和應用程式修訂版本所在的區域 (例如us-west-2)。如需區域的清單，請參閱中的區域和端點AWS一般參考。

若要設定環境變數，請從終端機呼叫下列項目：

```
export AWS_REGION=supported-region
```

其中 *supported-region* 為區域識別符 (例如 us-west-2)。

安裝 CodeDeploy 代理程式

- 對於 Ubuntu 伺服器內部部署執行個體，請遵循中的指示安裝 Ubuntu 伺服器的 CodeDeploy 代理程式，然後返回此頁面。
- 對於 RHEL 內部部署執行個體，請遵循中的指示安裝 Amazon Linux 或 RHEL 的 CodeDeploy 代理程式，然後返回此頁面。
- 對於 Windows Server 內部部署執行個體，請遵循中的指示安裝 Windows 伺服器的 CodeDeploy 代理程式，然後返回此頁面。

步驟 5：使用註冊內部部署執行個體 CodeDeploy

此步驟中的指示，假設您正在從現場部署執行個體本身註冊現場部署執行個體。您可以從已安裝和設定的個別 AWS CLI 裝置或執行個體註冊內部部署執行個體。

使 AWS CLI 用註冊內部部署執行個體，以 CodeDeploy 便在部署中使用。

在您可以使用之前 AWS CLI，您需要在中步驟 3：將設定檔新增至內部部署執行個體建立之臨時工作階段認證的 ARN。例如，對於您指定為 AssetTag12010298EX 的執行個體：

```
arn:sts:iam::123456789012:assumed-role/CodeDeployInstanceRole/AssetTag12010298EX
```

呼叫 register-on-premises-instance 命令，並指定：

- 唯一識別現場部署執行個體的名稱 (使用 --instance-name 選項)。

  > **Important**
  >
  > 為了協助識別現場部署執行個體，特別是用於偵錯用途，我們強烈建議您指定現場部署執行個體的某些獨特特性名稱 (例如，STS 登入資料的 session-name 和序號，或內部資產識別符，如果適用)。如果您指定 MAC 位址做為名稱，請注意 MAC 位址包含 CodeDeploy 不允許的字元，例如冒號 (:)。針對允許使用的字元清單，請參閱 CodeDeploy 配額

- 您在步驟 1：建立內部部署執行個體將承擔的 IAM 角色中設定以對多個現場部署執行個體進行身分驗證的 IAM 工作階段 ARN。

例如：

```
aws deploy register-on-premises-instance --instance-name name-of-instance --iam-
session-arn arn:aws:sts::account-id:assumed-role/role-to-assume/session-name
```

其中：

- *name-of-instance*是您用來識別內部部署執行個體的名稱，例如AssetTag12010298EX。
- *account-id* 為您機構組織的 12 位數帳戶 ID，例如 111222333444。
- *role-to-assume*是您為執行個體建立的 IAM 角色名稱，例如CodeDeployInstanceRole。
- *session-name* 是您在 步驟 2：使用以下方式為個別執行個體產生臨時認證 AWS STS 指定的工作階段角色名稱。

步驟 6：標記內部部署執行個體

您可以使用 AWS CLI 或主 CodeDeploy 控台來標記內部部署執行個體。(CodeDeploy使用內部部署執行個體標記來識別部署目標在部署期間。)

若要標記現場部署執行個體 (CLI)

- 呼叫 add-tags-to-on-內部部署執行個體命令，指定：
  - 唯一識別現場部署執行個體的名稱 (使用 `--instance-names` 選項)。
  - 現場部署執行個體標籤金鑰的名稱，以及您想使用的標籤值 (使用`--tags`選項)。您必須同時指定名稱和值。 CodeDeploy 不允許只有值的內部部署執行個體標記。

    例如：

```
aws deploy add-tags-to-on-premises-instances --instance-names AssetTag12010298EX
 --tags Key=Name,Value=CodeDeployDemo-OnPrem
```

若要標記現場部署執行個體 (主控台)

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在瀏覽窗格中，展開 [部署]，然後選擇 [內部部署執行個體
3. 請從現場部署執行個體的清單中，選擇您想要標記的現場部署執行個體名稱。
4. 在標籤清單中，選擇或輸入的標籤金鑰或標籤值。在您輸入標籤金鑰及標籤值後，將顯示另一個資料列。您最多可重複此標籤 10 次。若要移除標籤，請選擇 Remove (移除)。
5. 新增標籤後，選擇 Update Tags (更新標籤)。

步驟 7：將應用程式修訂部署到內部部署執行個

您現在已準備好將應用程式修訂部署至已註冊和加上標籤的現場部署執行個體。

您可以將應用程式修訂部署到現場部署執行個體的方式，類似於將應用程式修訂部署到 Amazon EC2 執行個體。如需說明，請參閱使用建立部署 CodeDeploy。這些指示含有一個連接到先決條件的連結，包含建立應用程式、建立部署群組以及準備應用程式修改版。如果您需要簡單的範例應用程式修訂來部署，您可以建立一個，如教學課程：使用 CodeDeploy (Windows 伺服器、Ubuntu 伺服器或 RHEL) 將應用程式部署到內部部署執行個體 中的步驟 2：建立範例應用程式修訂所述。

> **Important**
>
> 如果您在建立以內部部署執行個體為目標的部署群組時重複使用 CodeDeploy 服務角色，則必須包括服務角色原則陳述式的一`Action`部分。如需詳細資訊，請參閱 步驟 2：建立服務角色 CodeDeploy。

步驟 8：追蹤內部部署執行個體的部署

在您將應用程式修訂部署至已註冊和加入標籤的現場部署執行個體後，您可以追蹤部署的進度。

您可以使用類似於追蹤 Amazon EC2 執行個體的部署方式來追蹤現場部署執行個體。如需說明，請參閱 檢視 CodeDeploy 部署詳情 。

使用註冊命令 (IAM 使用者 ARN) 註冊內部部署執行個體

> **Important**
>
> 不建議使用 IAM 使用者註冊執行個體，因為它使用靜態 (永久) 登入資料進行身份驗證。為了提高安全性，我們建議使用臨時登入資料註冊執行個體進行驗證。如需詳細資訊，請參閱 使用命 register-on-premises-instance 令 (IAM 工作階段 ARN) 註冊內部部署執行個體。

> **Important**
>
> 確保您有適當的計劃來輪替 IAM 使用者的存取金鑰 (永久登入資料)。如需詳細資訊，請參閱旋轉存取金鑰。

本節說明如何設定內部部署執行個體，並以最少的努力註冊並加上標記。 CodeDeploy 當您使用單一或小型現場部署執行個體機群時，register 命令最有用。只有在使用 IAM 使用者 ARN 驗證執行個體時，才能使用此register命令。您無法將register命令與 IAM 工作階段 ARN 搭配使用來進行身份驗證。

當您使用該register命令時，您可以 CodeDeploy 執行以下操作：

- 如果您未使用命令指定內部部署執行個體，請在 AWS Identity and Access Management 中建立 IAM 使用者。
- 將 IAM 使用者的登入資料儲存至內部部署執行個體設定檔。

- 使用註冊內部部署執行個體 CodeDeploy。
- 如果您將標籤指定為命令的一部分，請將標籤新增至現場部署執行個體。

> **Note**
>
> 該register-on-premises-instance命令是註冊命令的替代方法。如果您想要設定內部部署執行個體，並 CodeDeploy大部分自行註冊並加上標記，請使用register-on-premises-instance命令。該register-on-premises-instance命令還提供了使用 IAM 工作階段 ARN 註冊執行個體的選項，而不是 IAM 使用者 ARN。如果您擁有大量現場部署執行個體機群，此方法可提供很大優勢。具體而言，您可以使用單一 IAM 工作階段 ARN 驗證多個執行個體，而不必逐個為每個內部部署執行個體建立 IAM 使用者。如需詳細資訊，請參閱 使用命 register-on-premises-instance 令 (IAM 使用者 ARN) 註冊內部部署執行個體 及 使用命 register-on-premises-instance 令 (IAM 工作階段 ARN) 註冊內部部署執行個體。

主題

步驟 1：在內部部署執行個體 AWS CLI 上安裝和設定

1. AWS CLI 在內部部署執行個體上安裝。請遵循「使用者指南」中的「取得設定」 AWS CLI中的AWS Command Line Interface 指示進行。

   > **Note**
   >
   > CodeDeploy 1.7.19 版及更 AWS CLI 新版本提供用於使用內部部署執行個體的命令。如果您已 AWS CLI 經安裝了，請aws --version致電檢查其版本。

2. AWS CLI 在內部部署執行個體上設定。請遵循使用者指南中的〈配置〉 AWS CLI中的AWS Command Line Interface指示進行。

Important

在設定 AWS CLI (例如，透過呼叫aws configure命令) 時，請務必指定 IAM 使用者的秘密金鑰 ID 和秘密存取金鑰，該使用者除了中指定的 AWS 權限外，至少具有下列存取權限設定內部部署執行個體的前提。這可讓您在內部部署執行個體上下載並安裝 CodeDeploy 代理程式。存取許可看起來類似下述：

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:*",
        "iam:CreateAccessKey",
        "iam:CreateUser",
        "iam:DeleteAccessKey",
        "iam:DeleteUser",
        "iam:DeleteUserPolicy",
        "iam:ListAccessKeys",
        "iam:ListUserPolicies",
        "iam:PutUserPolicy",
        "iam:GetUser",
        "tag:getTagKeys",
        "tag:getTagValues",
        "tag:GetResources"
      ],
      "Resource" : "*"
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource" : [
        "arn:aws:s3:::aws-codedeploy-us-east-2/*",
        "arn:aws:s3:::aws-codedeploy-us-east-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-2/*",
        "arn:aws:s3:::aws-codedeploy-ca-central-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
```

```
        "arn:aws:s3:::aws-codedeploy-eu-west-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-3/*",
        "arn:aws:s3:::aws-codedeploy-eu-central-1/*",
        "arn:aws:s3:::aws-codedeploy-il-central-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-east-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-4/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-1/*",
        "arn:aws:s3:::aws-codedeploy-sa-east-1/*"
      ]
    }
  ]
}
```

> **Note**
>
> 如果您在嘗試存取先前顯示的其中一個 Amazon S3 儲存貯體時看到存取遭拒錯誤，請嘗試省略儲存貯體資源 ARN 的一/*部分，例如。`arn:aws:s3:::aws-codedeploy-sa-east-1`

第 2 步：調用註冊命令

針對此步驟，我們假設您正在從現場部署執行個體本身註冊現場部署執行個體。您也可以從已安裝和設定的個別 AWS CLI 裝置或執行個體註冊內部部署執行個體，如上述步驟所述。

使用 AWS CLI 來呼叫寄存器命令，指定：

- 唯一識別內部部署執行個體的名稱 CodeDeploy (使用`--instance-name`選項)。

> **Important**
>
> 為了協助在稍後識別現場部署執行個體，特別是用於偵錯用途，我們強烈建議您使用映射至現場部署執行個體的某些獨特特性名稱 (例如序號或某些唯一的內部資產識別符，如果適用)。如果您指定名稱的 MAC 位址，請注意 MAC 位址包含 CodeDeploy 不允許的字元，例如冒號 (:)。針對允許使用的字元清單，請參閱 CodeDeploy 配額

- 或者，您要與此內部部署執行個體建立關聯的現有 IAM 使用者的 ARN (使用選--iam-user-arn項)。若要取得 IAM 使用者的 ARN，請呼叫 get-user 命令，或在 IAM 主控台的「使用者」區段中選擇 IAM 使用者名稱，然後在「摘要」區段中找到「使用者 ARN」值。如果未指定此選項，則 CodeDeploy 會在您的 AWS 帳戶中代表您建立 IAM 使用者，並將其與內部部署執行個體建立關聯。

  > **Important**
  >
  > 如果您指定 --iam-user-arn 選項，則也必須手動建立現場部署執行個體組態檔案，如步驟 4：將設定檔新增至內部部署執行個體中所述。
  > 您只能將一個 IAM 使用者與一個內部部署執行個體建立關聯。嘗試將單一 IAM 使用者與多個內部部署執行個體建立關聯可能會導致錯誤、部署失敗到這些內部部署執行個體，或是部署到停留在永久擱置狀態的現場部署執行個體。

- 或者，一組現場部署執行個體標籤 (含--tags選項) CodeDeploy ，用於識別要部署的 Amazon EC2 執行個體集。使用 Key=*tag-key*,Value=*tag-value* 來指定每個標籤 (例如 Key=Name,Value=Beta Key=Name,Value=WestRegion)。如果未指定此選項，則不會註冊任何標籤。若要稍後註冊標籤，請呼叫 add-tags-to-on-premises- 執行個體指令。
- 您可以選擇性 AWS 地註冊內部部署執行個體的區域 CodeDeploy (使用--region選項)。這必須是「區域」中列出的其中一個支援區域，以及 AWS 一般參考(例如，us-west-2) 中的端點。如果未指定此選項，將使用與呼叫 IAM 使用者相關聯的預設 AWS 區域。

例如：

```
aws deploy register --instance-name AssetTag12010298EX --iam-user-
arn arn:aws:iam::444455556666:user/CodeDeployUser-OnPrem --tags
 Key=Name,Value=CodeDeployDemo-OnPrem --region us-west-2
```

register 命令會執行以下動作：

1. 如果未指定現有的 IAM 使用者，請建立 IAM 使用者，將必要的許可附加至該使用者，然後產生對應的秘密金鑰和秘密存取金鑰。內部部署執行個體將使用此 IAM 使用者及其許可和登入資料進行驗證並與之互動 CodeDeploy。
2. 使用註冊內部部署執行個體 CodeDeploy。
3. 如果有指定，則會將使用--tags選項指定的標記與已註冊的內部部署執行個體名稱產生關聯。CodeDeploy
4. 如果已建立 IAM 使用者，也會在呼叫register命令的相同目錄中建立所需的組態檔案。

如果此命令發生錯誤，會出現錯誤訊息，說明如何手動完成剩下的步驟。否則即會出現成功訊息，說明如何呼叫在下一個步驟中列出的 install 命令。

步驟 3：調用安裝命令

在內部部署執行個體中 AWS CLI ，使用呼叫 install 命令，並指定：

- 組態檔案的路徑 (使用 `--config-file` 選項)。
- (選用) 是否取代現場部署執行個體上已存在的組態檔案 (使用 `--override-config` 選項)。如果未指定，將不會取代現有的組態檔案。
- 您可以選擇性 AWS 地註冊內部部署執行個體的區域 CodeDeploy (使用`--region`選項)。這必須是「區域」中列出的其中一個支援區域，以及 AWS 一般參考(例如，us-west-2) 中的端點。如果未指定此選項，將使用與呼叫 IAM 使用者相關聯的預設 AWS 區域。
- (選擇性) 要從中安裝 CodeDeploy 代理程式的自訂位置 (使用`--agent-installer`選項)。此選項適用於安裝 CodeDeploy 不正式支援的 CodeDeploy 代理程式的自訂版本 (例如以中的CodeDeploy代理程式儲存庫為基礎的自訂版本 GitHub)。該值必須是包含以下任一項目的 Amazon S3 儲存貯體的路徑：
  - CodeDeploy 代理程式安裝指令集 (適用於 Linux 或 UNIX 作業系統，類似於中的CodeDeploy代理程式儲存庫中 GitHub的安裝檔案)。
  - CodeDeploy 代理程式安裝程式套件 (.msi) 檔案 (適用於 Windows 作業系統)。

  如果未指定此選項， CodeDeploy 將盡可能從自己的位置安裝正式支援的代理程式版本，該 CodeDeploy 代理程式版本與內部部署執行個體上的作業系統相容。

例如：

```
aws deploy install --override-config --config-file /tmp/codedeploy.onpremises.yml --
region us-west-2 --agent-installer s3://aws-codedeploy-us-west-2/latest/codedeploy-
agent.msi
```

install 命令會執行以下動作：

1. 檢查現場部署執行個體是否為 Amazon EC2 執行個體。如果是，則會顯示錯誤訊息。
2. 將內部部署執行個體組態檔案從執行個體上的指定位置複製到 CodeDeploy 代理程式預期尋找的位置，前提是檔案尚未在該位置。

   對於 Ubuntu 服務器和紅帽企業 Linux（ RHEL ）），這是`/etc/codedeploy-agent/conf/codedeploy.onpremises.yml`。

對於視窗伺服器，這是`C:\ProgramData\Amazon\CodeDeploy\conf.onpremises.yml`。

如果 `--override-config` 選項已指定，則會建立或覆寫該檔案。

3. 在內部部署執行個體上安裝 CodeDeploy 代理程式，然後啟動它。

步驟 4：將應用程式修訂部署到內部部署執行個

您現在已準備好將應用程式修訂部署至已註冊和加上標籤的現場部署執行個體。

您可以將應用程式修訂部署到現場部署執行個體的方式，類似於將應用程式修訂部署到 Amazon EC2 執行個體。如需說明，請參閱使用建立部署 CodeDeploy。這些指示會連結至必要條件，包括建立應用程式、建立部署群組和準備應用程式修訂。如果您需要簡單的範例應用程式修訂來部署，您可以建立一個，如教學課程：使用 CodeDeploy (Windows 伺服器、Ubuntu 伺服器或 RHEL) 將應用程式部署到內部部署執行個體 中的步驟 2：建立範例應用程式修訂所述。

> **Important**
>
> 如果您在建立以內部部署執行個體為目標的部署群組時重複使用現有的 CodeDeploy 服務角色，則必`Tag:get*`須包括服務角色原則陳述式的一`Action`部分。如需詳細資訊，請參閱 步驟 2：建立服務角色 CodeDeploy。

步驟 5：追蹤內部部署執行個體的部署

在您將應用程式修訂部署至已註冊和加入標籤的現場部署執行個體後，您可以追蹤部署的進度。

您可以使用類似於追蹤 Amazon EC2 執行個體的部署方式來追蹤現場部署執行個體。如需說明，請參閱檢視 CodeDeploy 部署詳情 。

如需更多選項，請參閱 管理內部部署執行個體作 CodeDeploy。

使用命 register-on-premises-instance 令 (IAM 使用者 ARN) 註冊內部部署執行個體

> **Important**
>
> 不建議使用 IAM 使用者註冊執行個體，因為它使用靜態 (永久) 登入資料進行身份驗證。為了提高安全性，我們建議使用臨時登入資料註冊執行個體進行驗證。如需詳細資訊，請參閱 使用命 register-on-premises-instance 令 (IAM 工作階段 ARN) 註冊內部部署執行個體。

> **Important**
>
> 確保您有適當的計劃來輪替 IAM 使用者的存取金鑰 (永久登入資料)。如需詳細資訊，請參閱旋轉存取鍵。

依照這些指示設定內部部署執行個體，並使用靜態 IAM 使用者登入資料進行身份驗證，並 CodeDeploy 大部分自行註冊和標記。

主題

步驟 1：為內部部署執行個體建立 IAM 使用者

建立內部部署執行個體用來驗證和互動的 IAM 使用者 CodeDeploy。

> **Important**
>
> 您必須為每個參與的內部部署執行個體建立個別的 IAM 使用者。如果您嘗試針對多個內部部署執行個體重複使用個別 IAM 使用者，您可能無法使用成功註冊或標記這些內部部署執行個體 CodeDeploy。部署現場部署執行個體也許可能會卡在永久擱置狀態或一起失敗。

我們建議您為 IAM 使用者指派一個識別其用途的名稱，例如 CodeDeployUser-OnPrem。

您可以使用 AWS CLI 或 IAM 主控台建立 IAM 使用者。如需詳細資訊，請參閱在您的 AWS 帳戶中建立 IAM 使用者。

> **Important**
>
> 無論您是使用 AWS CLI 或 IAM 主控台建立新的 IAM 使用者，請記下為使用者提供的使用者 ARN。您之後將需要用到這個資訊 步驟 4：將設定檔新增至內部部署執行個體 和 步驟 8：使用註冊內部部署執行個體 CodeDeploy。

步驟 2：將許可指派給 IAM 使用者

如果您的現場部署執行個體將從 Amazon S3 儲存貯體部署應用程式修訂，則必須將許可指派給 IAM 使用者，以便與這些儲存貯體互動。您可以使用 AWS CLI 或 IAM 主控台指派許可。

> **Note**
>
> 如果您將只從 GitHub 儲存庫部署應用程式修訂版本，請略過此步驟並直接移至步驟 3：取得 IAM 使用者登入資料。您仍然需要在中建立的 IAM 使用者的相關資訊步驟 1：為內部部署執行個體建立 IAM 使用者。 它將在後面的步驟中使用。)

指派許可權限 (CLI)。

1. 在您用來呼叫的 Amazon EC2 執行個體或裝置上建立包含下列政策內容的檔案 AWS CLI。為檔案命名如 **CodeDeploy-OnPrem-Permissions.json**，然後儲存檔案。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 建議您將此政策限制為只有現場部署執行個體需要存取的 Amazon S3 儲存貯體。如果限制此政策，請確保還授予對包含 AWS CodeDeploy 代理程式之 Amazon S3 儲存貯體的存取權。否則，每當在關聯的內部部署執行個體上安裝或更新 CodeDeploy 代理程式時，就可能會發生錯誤。
>
> 例如：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3:::replace-with-your-s3-bucket-name/*",
        "arn:aws:s3:::aws-codedeploy-us-east-2/*",
        "arn:aws:s3:::aws-codedeploy-us-east-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-2/*",
        "arn:aws:s3:::aws-codedeploy-ca-central-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-3/*",
        "arn:aws:s3:::aws-codedeploy-eu-central-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-central-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-north-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-south-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-south-2/*",
        "arn:aws:s3:::aws-codedeploy-il-central-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-east-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-3/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-3/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-4/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-1/*",
```

```
        "arn:aws:s3:::aws-codedeploy-ap-south-2/*",
        "arn:aws:s3:::aws-codedeploy-me-central-1/*",
        "arn:aws:s3:::aws-codedeploy-me-south-1/*",
        "arn:aws:s3:::aws-codedeploy-sa-east-1/*"
      ]
    }
  ]
}
```

2. 呼叫命put-user-policy令，指定 IAM 使用者的名稱 (使用--user-name選項)、原則名稱 (使用--policy-name選項)，以及新建立的政策文件的路徑 (使用選--policy-document項)。例如，假設 **CodeDeploy-OnPrem-Permissions.json** 檔案與您正呼叫的命令位於相同的目錄 (資料夾)：

> **Important**
>
> 請確認在檔案名稱之前包含 file://。這是此命令必要項目。

```
aws iam put-user-policy --user-name CodeDeployUser-OnPrem --policy-name CodeDeploy-OnPrem-Permissions --policy-document file://CodeDeploy-OnPrem-Permissions.json
```

指派許可權限 (主控台)

1. 在 https://console.aws.amazon.com/iam/ 中開啟 IAM 主控台。
2. 在導覽窗格中，選擇 Policies (政策)，然後選擇 Create Policy (建立政策)。(出現 Get Started (開始使用) 按鈕時先選擇它，然後選擇 Create Policy (建立政策)。)
3. 在建立您自己的政策旁邊，選擇選取。
4. 在 政策名稱 方塊中，輸入此政策的名稱。(例如，**CodeDeploy-OnPrem-Permissions**)。
5. 在「政策文件」方塊中，輸入或貼上下列許可表示式，以 AWS CodeDeploy 便代表 IAM 使用者將政策中指定的任何 Amazon S3 儲存貯體的應用程式修訂部署到現場部署執行個體：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
```

```
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

6. 選擇建立政策。
7. 在導覽窗格中，選擇使用者 。
8. 在使用者清單中，瀏覽並選擇您在其中建立的 IAM 使用者的名稱步驟 1：為內部部署執行個體建立 IAM 使用者。
9. 在 Permissions (許可) 標籤上，Managed Policies (受管政策) 中，選擇 Attach Policy (連接政策)。
10. 選擇政策的名稱 **CodeDeploy-OnPrem-Permissions**，然後選擇 Attach Policy (附加政策)。

步驟 3：取得 IAM 使用者登入資料

取得 IAM 使用者的秘密金鑰 ID 和秘密存取金鑰。您將需要使用他們於 步驟 4：將設定檔新增至內部部署執行個體。您可以使用 AWS CLI 或 IAM 主控台取得秘密金鑰 ID 和秘密存取金鑰。

> **Note**
>
> 如果您已經有私密金鑰 ID 和私密存取金鑰，請略過此步驟並直接前往 步驟 4：將設定檔新增至內部部署執行個體。
> 如果使用者想要與 AWS 之外的 AWS Management Console. 授與程式設計存 AWS取權的方式取決於正在存取的使用者類型。
> 若要授與使用者程式設計存取權，請選擇下列其中一個選項。

哪個使用者需要程式設計存取權？	到	By
人力身分 (IAM Identity Center 中管理的使用者)	使用臨時登入資料來簽署對 AWS CLI、 AWS SDK 或 AWS API 的程式設計要求。	請依照您要使用的介面所提供的指示操作。 • 如需詳細資訊 AWS CLI，請參閱《使 AWS CLI

哪個使用者需要程式設計存取權？	到	By
		用AWS Command Line Interface者指南》 AWS IAM Identity Center中的〈配置使用〉。 • 如需 AWS SDK、工具和 AWS API，請參閱 AWS SDK 和工具參考指南中的 IAM 身分中心身分驗證。
IAM	使用臨時登入資料來簽署對 AWS CLI、 AWS SDK 或 AWS API 的程式設計要求。	遵循《IAM 使用者指南》中的〈將臨時登入資料搭配 AWS 資源使用〉中的指示
IAM	(不建議使用) 使用長期認證簽署對 AWS CLI、 AWS SDK 或 AWS API 的程式設計要求。	請依照您要使用的介面所提供的指示操作。 • 如需相關資訊 AWS CLI，請參閱使用指南中的使用 IAM 使用者登入資料進行驗證。AWS Command Line Interface • 對於 AWS SDK 和工具，請參閱 AWS SDK 和工具參考指南中的使用長期憑據進行身份驗證。 • 如需 AWS API，請參閱 IAM 使用者指南中的管理 IAM 使用者的存取金鑰。

取得登入資料 (CLI)

1. 呼叫命list-access-keys令，指定 IAM 使用者的名稱 (使用`--user-name`選項)，並僅查詢存取金鑰 ID (使用`--query`和`--output`選項)。例如：

```
aws iam list-access-keys --user-name CodeDeployUser-OnPrem --query
 "AccessKeyMetadata[*].AccessKeyId" --output text
```

2. 如果輸出中沒有出現任何金鑰，或者輸出中只顯示一個金鑰的相關資訊，請呼叫create-access-key命令，指定 IAM 使用者的名稱 (使用--user-name選項)：

```
aws iam create-access-key --user-name CodeDeployUser-OnPrem
```

在呼叫輸出的 create-access-key 命令中，備註 AccessKeyId 的值與 SecretAccessKey 欄位。您將需要這個資訊步驟 4：將設定檔新增至內部部署執行個體。

> **Important**
>
> 這是唯一您可以存取此私密存取金鑰的時間。如果您忘記或遺失存取此私密存取金鑰，則您需要產生新的私密存取金鑰，請遵循 步驟 3：取得 IAM 使用者登入資料 中的步驟執行。

3. 如果已列出兩個存取金鑰，您必須透過呼叫delete-access-key命令、指定 IAM 使用者的名稱 (使用--user-name選項) 以及要刪除的存取金鑰的 ID (使用--access-key-id選項) 來刪除其中一個。接著呼叫 create-access-key 命令，如此步驟中先前所述。以下範例呼叫 delete-access-key 命令：

```
aws iam delete-access-key --user-name CodeDeployUser-OnPrem --access-key-id access-key-ID
```

> **Important**
>
> 如果您呼叫delete-access-key命令刪除其中一個存取金鑰，且內部部署執行個體已如中所述使用此存取金鑰步驟 4：將設定檔新增至內部部署執行個體，則需要步驟 4：將設定檔新增至內部部署執行個體再次遵循中的指示，以指定與此 IAM 使用者相關聯的不同存取金鑰 ID 和秘密存取金鑰。其他，任何部署到現場部署執行個體可能卡在永久擱置狀態或一起失敗。

如何取得登入資料 (主控台)

1. a. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。

b. 如果未顯示於使用者清單，於導覽窗格請選擇 Users (使用者)。

c. 在使用者清單中，瀏覽並選擇您在其中建立的 IAM 使用者的名稱步驟 1：為內部部署執行個體建立 IAM 使用者。

2. 在 Security credentials (安全登入資料) 標籤，如果沒有金鑰或只有列出一個金鑰，請選擇 Create access key (建立存取金鑰)。

   如果列出兩個存取金鑰，則您必須刪除其中一個。選擇其中一個存取金鑰旁的 Delete (刪除)，然後選擇 Create access key (建立存取金鑰)。

   > **Important**
   >
   > 如果您選擇其中一個存取金鑰旁邊的 [刪除]，且內部部署執行個體已如中所述使用此存取金鑰步驟 4：將設定檔新增至內部部署執行個體，則需要步驟 4：將設定檔新增至內部部署執行個體再次依照中的指示指定與此 IAM 使用者相關聯的不同存取金鑰 ID 和秘密存取金鑰。否則，部署到現場部署執行個體可能卡在永久擱置狀態或一起失敗。

3. 選擇 顯示 和備註的存取金鑰 ID 和私密存取金鑰。下一個步驟您將需要這個資訊。或者，您可以選擇 下載 .csv 檔案，儲存存取金鑰ID以及秘密存取金鑰的副本。

   > **Important**
   >
   > 除非您註記或下載登入資料，否則這將是唯一一次您可以存取到此祕密存取金鑰的機會。如果您忘記或遺失存取此私密存取金鑰，則您需要產生新的私密存取金鑰，請遵循 步驟 3：取得 IAM 使用者登入資料 中的步驟執行。

4. 選擇 Close (關閉) 傳回給 使用者 > ***IAM #####*** 頁面。

步驟 4：將設定檔新增至內部部署執行個體

使用 root 或管理員許可，將組態檔案新增至現場部署執行個體。此設定檔將用於宣告 IAM 使用者登入資料和要用於的目標 AWS 區域 CodeDeploy。該檔案必須新增至現場部署執行個體上的特定位置。檔案必須包含 IAM 使用者的 ARN、秘密金鑰 ID、秘密存取金鑰和目標 AWS 區域。這個檔案必須遵循特定的格式。

1. 在內部部署執行個體的下列位置建立名為 `codedeploy.onpremises.yml` (針對 Ubuntu 伺服器或 RHEL 內部部署執行個體) 或 `conf.onpremises.yml` (適用於 Windows Server 內部部署執行個體) 的檔案：

- 對於 Ubuntu 服務器：`/etc/codedeploy-agent/conf`
- 對於視窗伺服器：`C:\ProgramData\Amazon\CodeDeploy`

2. 使用文字編輯器，將下列資訊新增至新建立的 `codedeploy.onpremises.yml` 或 `conf.onpremises.yml` 檔案：

```
---
aws_access_key_id: secret-key-id
aws_secret_access_key: secret-access-key
iam_user_arn: iam-user-arn
region: supported-region
```

其中：

- *secret-key-id*是您在步驟 1：為內部部署執行個體建立 IAM 使用者或中註明的對應 IAM 使用者的秘密金鑰 ID 步驟 3：取得 IAM 使用者登入資料。
- *secret-access-key*是您在步驟 1：為內部部署執行個體建立 IAM 使用者或中註明的對應 IAM 使用者的秘密存取金鑰步驟 3：取得 IAM 使用者登入資料。
- *iam-user-arn*是您先前在中步驟 1：為內部部署執行個體建立 IAM 使用者提到的 IAM 使用者的 ARN。
- 支援*##*是您的 CodeDeploy 應用程式、部署群組和應用程式修訂版所 CodeDeploy 支援的區域識別碼 (例如)。`us-west-2`如需區域的清單，請參閱中的區域和端點AWS 一般參考。

> **Important**
>
> 如果您選擇中其中一個存取金鑰旁邊的 [刪除]步驟 3：取得 IAM 使用者登入資料，而您的內部部署執行個體已在使用相關聯的存取金鑰 ID 和秘密存取金鑰，則必須依照中步驟 4：將設定檔新增至內部部署執行個體的指示指定與此 IAM 使用者相關聯的不同存取金鑰 ID 和秘密存取金鑰。其他，任何部署到您的現場部署執行個體可能卡在永久擱置狀態或一起失敗。

步驟 5：安裝和配置 AWS CLI

在內部部署執行個體 AWS CLI 上安裝和設定。(AWS CLI 將在中使用，在步驟 7：安裝 CodeDeploy 代理程式 內部部署執行個體上下載並安裝 CodeDeploy 代理程式。)

1. 若要在內部部署執行個體 AWS CLI 上安裝，請依照AWS Command Line Interface 使用者指南中的 [取得設定] AWS CLI 中的指示進行。

   > **Note**
   >
   > CodeDeploy 用於使用內部部署執行個體的命令在. AWS CLI如果您已 AWS CLI 經安裝了版本，則可以通過調用來檢查其版本aws --version。

2. 若要在內部部署執行個體 AWS CLI 上設定，請依照AWS Command Line Interface 使用者指南中的〈設定〉 AWS CLI中的指示進行。

   > **Important**
   >
   > 設定 AWS CLI (例如，透過呼叫aws configure命令) 時，請務必指定 IAM 使用者的秘密金鑰 ID 和秘密存取金鑰，該使用者除了中指定的 AWS 存取權限之外，至少具有下列存取權限設定內部部署執行個體的前提。這可讓您在內部部署執行個體上下載並安裝 CodeDeploy 代理程式：

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:*"
      ],
      "Resource" : "*"
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource" : [
        "arn:aws:s3:::aws-codedeploy-us-east-2/*",
        "arn:aws:s3:::aws-codedeploy-us-east-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-2/*",
        "arn:aws:s3:::aws-codedeploy-ca-central-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
```

```
        "arn:aws:s3:::aws-codedeploy-eu-west-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-3/*",
        "arn:aws:s3:::aws-codedeploy-eu-central-1/*",
        "arn:aws:s3:::aws-codedeploy-il-central-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-east-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-4/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-1/*",
        "arn:aws:s3:::aws-codedeploy-sa-east-1/*"
      ]
    }
  ]
}
```

這些存取權限可以指派給您在其中建立的 IAM 使用者，步驟 1：為內部部署執行個體建立 IAM 使用者或指派給其他 IAM 使用者。若要將這些許可指派給 IAM 使用者，請遵循中的指示步驟 1：為內部部署執行個體建立 IAM 使用者，使用這些存取權限，而不是該步驟中的指示。

步驟 6：設定 AW_ 區域環境變數 (僅限 Ubuntu 伺服器和 RHEL)

如果您沒有在內部部署執行個體上執行 Ubuntu 伺服器或 RHEL，請略過此步驟並直接前往。步驟 7：安裝 CodeDeploy 代理程式

在 Ubuntu 伺服器或 RHEL 內部部署執行個體上安裝 CodeDeploy 代理程式，並讓執行個體在有新版本可用時更新 CodeDeploy 代理程式。您可以透過將執行個體上的AWS_REGION環境變數設定為其中一個支援區域的識別碼來執行此操作 CodeDeploy。建議您將值設定為 CodeDeploy 應用程式、部署群組和應用程式修訂版本所在的區域 (例如us-west-2)。如需區域的清單，請參閱中的區域和端點AWS一般參考。

若要設定環境變數，請從終端機呼叫下列項目：

```
export AWS_REGION=supported-region
```

其中 *supported-region* 為區域識別符 (例如 us-west-2)。

步驟 7：安裝 CodeDeploy 代理程式

在內部部署執行個體上安裝 CodeDeploy 代理程式：

- 對於 Ubuntu 伺服器內部部署執行個體，請遵循中的指示安裝 Ubuntu 伺服器的 CodeDeploy 代理程式，然後返回此頁面。
- 對於 RHEL 內部部署執行個體，請遵循中的指示安裝 Amazon Linux 或 RHEL 的 CodeDeploy 代理程式，然後返回此頁面。
- 對於 Windows Server 內部部署執行個體，請遵循中的指示安裝 Windows 伺服器的 CodeDeploy 代理程式，然後返回此頁面。

步驟 8：使用註冊內部部署執行個體 CodeDeploy

此步驟中的指示，假設您正在從現場部署執行個體本身註冊現場部署執行個體。您可以從已安裝和設定的個別 AWS CLI 裝置或執行個體註冊內部部署執行個體，如中所述步驟 5：安裝和配置 AWS CLI。

使 AWS CLI 用註冊內部部署執行個體，以 CodeDeploy 便在部署中使用。

1. 您需要在步驟 1：為內部部署執行個體建立 IAM 使用者中建立之 IAM 使用者的使用者 ARN AWS CLI，才能使用。如果您還沒有使用者 ARN，請呼叫 get-user 命令，指定 IAM 使用者的名稱 (使用`--user-name`選項)，然後只查詢使用者 ARN (使用和選項)：`--query``--output`

   ```
   aws iam get-user --user-name CodeDeployUser-OnPrem --query "User.Arn" --output text
   ```

2. 呼叫 register-on-premises-instance 命令，並指定：

   - 唯一識別現場部署執行個體的名稱 (使用 `--instance-name` 選項)。

     > **Important**
     >
     > 為了幫助鑑別現場部署執行個體，特別是偵錯程序，我們強烈建議您指定一個名稱，其對應到一些現場部署執行個體的獨特字元 (例如，序列數字或一個內部資產鑑別者，若適用的話)。如果您指定 MAC 位址做為名稱，請注意 MAC 位址包含 CodeDeploy 不允許的字元，例如冒號 (`:`)。針對允許使用的字元清單，請參閱 CodeDeploy 配額

   - 您在中建立之 IAM 使用者的使用者 ARN 步驟 1：為內部部署執行個體建立 IAM 使用者 (使用選`--iam-user-arn`項)。

     例如：

```
aws deploy register-on-premises-instance --instance-name AssetTag12010298EX --
iam-user-arn arn:aws:iam::444455556666:user/CodeDeployUser-OnPrem
```

步驟 9：標記內部部署執行個體

您可以使用 AWS CLI 或主 CodeDeploy 控台來標記內部部署執行個體。(CodeDeploy使用內部部署執行個體標記來識別部署目標在部署期間。)

若要標記現場部署執行個體 (CLI)

- 呼叫 add-tags-to-on-內部部署執行個體命令，指定：

  - 唯一識別現場部署執行個體的名稱 (使用 `--instance-names` 選項)。
  - 現場部署執行個體標籤金鑰的名稱，以及您想使用的標籤值 (使用`--tags`選項)。您必須同時指定名稱和值。 CodeDeploy 不允許只有值的內部部署執行個體標記。

    例如：

```
aws deploy add-tags-to-on-premises-instances --instance-names AssetTag12010298EX
 --tags Key=Name,Value=CodeDeployDemo-OnPrem
```

若要標記現場部署執行個體 (主控台)

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 從 CodeDeploy 功能表中選擇內部部署執行個體。
3. 在現場部署執行個體的清單中，選擇箭號到下一個您想標籤的內部部署執行個體。
4. 在標籤清單中，選擇或輸入的標籤金鑰或標籤值。在您輸入標籤金鑰及標籤值後，將顯示另一個資料列。您最多可重複此標籤 10 次。若要移動標籤，請選擇刪除圖示 ( )。

5. 新增標籤後，選擇 Update Tags (更新標籤)。

步驟 10：將應用程式修訂部署到內部部署執行個

您現在已準備好將應用程式修訂部署至已註冊和加上標籤的現場部署執行個體。

您可以將應用程式修訂部署到現場部署執行個體的方式，類似於將應用程式修訂部署到 Amazon EC2 執行個體。如需說明，請參閱使用建立部署 CodeDeploy。這些指示含有一個連接到先決條件的連結，包含建立應用程式、建立部署群組以及準備應用程式修改版。如果您需要簡單的範例應用程式修訂來部署，您可以建立一個，如教學課程：使用 CodeDeploy (Windows 伺服器、Ubuntu 伺服器或 RHEL) 將應用程式部署到內部部署執行個體 中的步驟 2：建立範例應用程式修訂所述。

> **Important**
>
> 如果您在建立以內部部署執行個體為目標的部署群組時重複使用 CodeDeploy 服務角色，則必須包括服務角色原則陳述式的一`Action`部分。如需詳細資訊，請參閱 步驟 2：建立服務角色 CodeDeploy。

步驟 11：追蹤內部部署執行個體的部署

在您將應用程式修訂部署至已註冊和加入標籤的現場部署執行個體後，您可以追蹤部署的進度。

您可以使用類似於追蹤 Amazon EC2 執行個體的部署方式來追蹤現場部署執行個體。如需說明，請參閱 檢視 CodeDeploy 部署詳情 。

管理內部部署執行個體作 CodeDeploy

在您註冊內部部署執行個體之後，請遵循本節中的指示來管理作業 CodeDeploy，例如取得更多相關資訊、從中移除標記，以及解除安裝和取消註冊內部部署執行個體。

主題

- 取得單一內部部署執行個體的資訊
- 取得多個內部部署執行個體的
- 手動移除內部部署執行個體標記
- 自動解除安裝 CodeDeploy 代理程式，並從內部部署執行個體移除設定檔
- 自動註銷內部部署執行個體

- 手動取消註冊內部部署執行個

取得單一內部部署執行個體的資訊

您可以在遵照 檢視 CodeDeploy 部署詳情 的指示取得單一現場部署執行個體的資訊。您可以使用 AWS CLI 或主 CodeDeploy 控台取得單一內部部署執行個體的詳細資訊。

取得有關單一現場部署執行個體 (CLI) 的資訊

- 呼叫命get-on-premises-instance令，指定唯一識別內部部署執行個體的名稱 (使用--instance-name選項)：

```
aws deploy get-on-premises-instance --instance-name AssetTag12010298EX
```

取得單一現場部署執行個體的相關資訊 (主控台)

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在瀏覽窗格中，展開 [部署]，然後選擇 [內部部署執行個體
3. 在現場部署執行個體清單中，請選擇現場部署執行個體的名稱，以查看其詳細資訊。

取得多個內部部署執行個體的

您可以在遵照 檢視 CodeDeploy 部署詳情 的指示取得有關現場部署執行個體的資訊。您可以使用 AWS CLI 或主 CodeDeploy 控台取得有關內部部署執行個體的詳細資訊。

為取得有關多個現場部署執行個體 (CLI) 的資訊

1. 如需內部部署執行個體名稱的清單，請呼叫list-on-premises-instances命令，並指定：
   - 取得所有註冊或撤銷註冊的現場部署執行個體的資訊 (分別依序使用 `--registration-status` 選項和 `Registered` 或 `Deregistered`)。如果您省略此步驟，則將傳回註冊和撤銷註冊現場部署執行個體的名稱。

- 取得僅有特定現場部署執行個體標籤的現場部署執行個體 (加上 `--tag-filters` 選項)。對於每個現場部署執行個體標籤，請指定 `Key`、`Value`以及 `Type` (請務必為此 `KEY_AND_VALUE`)。在每一個 `Key`、`Value` 以及 `Type` 三者之間，使用空格分隔多個現場部署執行個體標籤。

  例如：

```
aws deploy list-on-premises-instances --registration-status Registered
 --tag-filters Key=Name,Value=CodeDeployDemo-OnPrem,Type=KEY_AND_VALUE
 Key=Name,Value=CodeDeployDemo-OnPrem-Beta,Type=KEY_AND_VALUE
```

2. 如需詳細資訊，請使用內部部署batch-get-on-premises執行個體的名稱呼叫- instances 命令 (使用選`--instance-names`項)：

```
aws deploy batch-get-on-premises-instances --instance-names AssetTag12010298EX
 AssetTag09920444EX
```

取得多個現場部署執行個體的相關資訊 (主控台)

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在瀏覽窗格中，展開 [部署]，然後選擇 [內部部署執行個體

   系統會隨即顯示該現場部署執行個體的相關資訊。

手動移除內部部署執行個體標記

一般而言，從現場部署執行個體移除您不再使用的現場部署執行個體標籤，或從部署群組移除依賴該標籤的現場部署執行個體。您可以使用 AWS CLI 或 AWS CodeDeploy 主控台從內部部署執行個體移除內部部署執行個體標記。

在您撤銷註冊前，不需要從現場部署執行個體移除現場部署執行個體標籤。

從現場部署執行個體手動移除現場部署執行個體標籤並不會撤銷註冊該執行個體。它不會從執行個體解除安裝 CodeDeploy 代理程式。它不會從執行個體移除組態檔案。它不會刪除與執行個體相關聯的 IAM 使用者。

若要自動撤銷註冊現場部署執行個體，請參閱 自動註銷內部部署執行個體。

若要手動撤銷註冊現場部署執行個體，請參閱 手動取消註冊內部部署執行個。

若要自動解除安裝 CodeDeploy 代理程式並從內部部署執行個體移除組態檔案，請參閱自動解除安裝 CodeDeploy 代理程式，並從內部部署執行個體移除設定檔。

若只要從內部部署執行個體手動解除安裝 CodeDeploy 代理程式，請參閱管理 CodeDeploy 代理程式作。

若要手動刪除相關的 IAM 使用者，請參閱從您的 AWS 帳戶刪除 IAM 使用者。

若要從現場部署執行個體 (CLI) 移除現場部署執行個體標籤

- 呼叫 remove-tags-from-on-內部部署執行個體，指定：

  - 唯一識別現場部署執行個體的名稱 (使用 `--instance-names` 選項)。
  - 您要移除的標籤名稱與標籤數值 (使用 `--tags` 選項)。

    例如：

    ```
    aws deploy remove-tags-from-on-premises-instances --instance-names
     AssetTag12010298EX --tags Key=Name,Value=CodeDeployDemo-OnPrem
    ```

從現場部署執行個體移除現場部署執行個體標籤 (主控台)

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在瀏覽窗格中，展開 [部署]，然後選擇 [內部部署執行個體
3. 在現場部署執行個體清單，請選擇要移除標籤的現場部署執行個體名稱。

4. 在 Tags (標籤) 區段，在每一個您想要移除的標籤旁選擇 Remove (移除)。
5. 刪除這些標籤後，請選擇 Update tags (更新標籤)。

自動解除安裝 CodeDeploy 代理程式，並從內部部署執行個體移除設定檔

一般而言，當您不再打算部署到內部部署執行個體之後，您可以解除安裝 CodeDeploy 代理程式，並從內部部署執行個體移除設定檔。

> **Note**
>
> 自動解除安裝 CodeDeploy 代理程式並從內部部署執行個體移除設定檔並不會取消註冊內部部署執行個體。它不會取消任何與現場部署執行個體關聯的現場部署執行個體標籤。它不會刪除與內部部署執行個體相關聯的 IAM 使用者。
> 若要自動撤銷註冊現場部署執行個體，請參閱 自動註銷內部部署執行個體。
> 若要手動撤銷註冊現場部署執行個體，請參閱 手動取消註冊內部部署執行個。
> 若要手動取消任何關聯現場部署執行個體標籤，請參閱 手動移除內部部署執行個體標記。
> 若要從內部部署執行個體手動解除安裝 CodeDeploy 代理程式，請參閱 管理 CodeDeploy 代理程式作
> 若要手動刪除相關的 IAM 使用者，請參閱從您的 AWS 帳戶刪除 IAM 使用者。

在內部部署執行個體中，使 AWS CLI 用呼叫解除安裝命令。

例如：

```
aws deploy uninstall
```

uninstall 命令會執行以下動作：

1. 停止內部部署執行個體上執行的 CodeDeploy 代理程式。
2. 從內部部署執行個體解除安裝 CodeDeploy 代理程式。
3. 從現場部署執行個體移除組態檔案。（對於 Ubuntu 服務器和 RHEL，這是`/etc/codedeploy-agent/conf/codedeploy.onpremises.yml`。 對於視窗伺服器，這是`C:\ProgramData\Amazon\CodeDeploy\conf.onpremises.yml`。）

自動註銷內部部署執行個體

一般而言，不再規劃用於部署的現場部署執行個體，您會撤銷其註冊。當您註銷現場部署執行個體的註冊，即使現場部署執行個體可能為部署群組的現場部署執行個體標籤的一部分，該現場部署執行個體還是不會包含於任何部署裡。您可以使用取 AWS CLI 消註冊內部部署執行個體。

> **Note**
>
> 您無法使用 CodeDeploy 主控台取消註冊內部部署執行個體。此外，將內部部署執行個體撤銷註冊，會移除與內部部署執行個體相關聯的內部部署執行個體標籤。它不會從內部部署執行個體解除安裝 CodeDeploy 代理程式。它不會從現場部署執行個體移除現場部署執行個體的組態檔案。
>
> 若要使用 CodeDeploy 主控台執行本節中部分 (但不是全部) 活動，請參閱的 CodeDeploy 主控台一節手動取消註冊內部部署執行個。
>
> 若要手動取消任何關聯現場部署執行個體標籤，請參閱 手動移除內部部署執行個體標記。
>
> 若要自動解除安裝 CodeDeploy 代理程式並從內部部署執行個體移除組態檔案，請參閱自動解除安裝 CodeDeploy 代理程式，並從內部部署執行個體移除設定檔。
>
> 若要僅從內部部署執行個體手動解除安裝 CodeDeploy 代理程式，請參閱管理 CodeDeploy 代理程式作。

使用呼叫 AWS CLI 取消註冊命令，指定：

- 唯一識別內部部署執行個體的名稱 CodeDeploy (使用`--instance-name`選項)。
- 選擇性地刪除與內部部署執行個體相關聯的 IAM 使用者。預設行為是刪除 IAM 使用者。如果您不想刪除內部部署執行個體關聯的 IAM 使用者，請在命令中指定 `--no-delete-iam-user` 選項。
- 選擇性 AWS 地指出現場部署執行個體註冊的地區 CodeDeploy (可`--region`選擇性)。這必須是「區域」中列出的其中一個支援區域，以及 AWS 一般參考(例如`us-west-2`) 中的端點。如果未指定此選項，將使用與呼叫 IAM 使用者相關聯的預設 AWS 區域。

將執行個體撤銷註冊並刪除使用者的範例：

```
aws deploy deregister --instance-name AssetTag12010298EX --region us-west-2
```

將執行個體撤銷註冊並且不會刪除使用者的範例：

```
aws deploy deregister --instance-name AssetTag12010298EX --no-delete-iam-user --region
 us-west-2
```

deregister 命令會執行以下動作：

1. 使用取消註冊內部部署執行個體 CodeDeploy。
2. 如有指定，則刪除與內部部署執行個體相關聯的 IAM 使用者。

一旦取消註冊現場部署執行個體，將導致以下情況：

- 主控台不會再顯示該執行個體。
- 您能立即建立具有相同名稱的其他執行個體。

如果此命令發生錯誤，會出現錯誤訊息，說明如何手動完成剩下的步驟。不然的話，則會出現成功訊息，說明如何呼叫 uninstall 命令。

手動取消註冊內部部署執行個

一般而言，不再規劃用於部署的現場部署執行個體，您會撤銷其註冊。您可以使用手動 AWS CLI 取消註冊內部部署執行個體。

手動取消註冊內部部署執行個體不會解除安裝 CodeDeploy 代理程式。它不會從執行個體移除組態檔案。它不會刪除與執行個體相關聯的 IAM 使用者。它不會移除與執行個體相關的任何標籤。

若要自動解除安裝 CodeDeploy 代理程式並從內部部署執行個體移除組態檔案，請參閱自動解除安裝 CodeDeploy 代理程式，並從內部部署執行個體移除設定檔。

如果只要手動解除安裝 CodeDeploy 代理程式，請參閱管理 CodeDeploy 代理程式作。

若要手動刪除相關的 IAM 使用者，請參閱從您的 AWS 帳戶刪除 IAM 使用者。

若僅要手動移除任何相關的現場部署執行個體標籤，請參閱 手動移除內部部署執行個體標記。

- 呼叫命deregister-on-premises-instance令，指定唯一識別內部部署執行個體的名稱 (使用`--instance-name`選項)：

  ```
  aws deploy deregister-on-premises-instance --instance-name AssetTag12010298EX
  ```

  一旦取消註冊現場部署執行個體，將導致以下情況：

- 主控台不會再顯示該執行個體。
- 您能立即建立具有相同名稱的其他執行個體。

查看實例詳細信息 CodeDeploy

您可以使用 CodeDeploy 主控台 AWS CLI、或 CodeDeploy API 來檢視部署中使用的執行個體的詳細資料。

如需使用 CodeDeploy API 動作檢視執行個體的詳細資訊 GetDeploymentInstance，請參閱ListDeploymentInstances、和ListOnPremisesInstances。

主題

查看實例詳細信息（控制台）

若要檢視執行個體的詳細資訊，請執行以下步驟：

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在瀏覽窗格中，展開 [部署]，然後選擇 [部署]。

   > **Note**
   >
   > 如果未顯示任何項目，請確定已選取正確的區域。在導覽列的區域選取器中，選擇「區域」和「端點」中列出的其中一個區域AWS 一般參考。 CodeDeploy 僅在這些地區支援。

3. 若要顯示部署詳細資訊，請選擇執行個體的部署 ID。
4. 您可以在部署頁面的 Instance activity (執行個體活動) 區段檢視所有執行個體。

5. 若要查看執行個體個別部署生命週期事件的相關資訊，請移至部署詳細資訊頁面上的 Events (事件) 欄位，然後選擇 View events (檢視事件)。

> **Note**
>
> 如果 失敗 顯示在任何生命週期事件中，在執行個體詳細資訊頁面上，選擇 檢視記錄檔，在 EC2 裡檢視，或在兩者上檢視。如需疑難排解的秘訣，請參閱對執行個體問題進行故障診斷。

6. 如果您想查看有關 Amazon EC2 執行個體的詳細資訊，請在「執行個體 ID」欄中選擇執行個體的 ID。

檢視執行個體詳細資訊 (CLI)

若要使用 AWS CLI 檢視執行個體詳細資訊，請呼叫`get-deployment-instance`命令或命`list-deployment-instances`令。

若要檢視單一執行個體的詳細資訊，請呼叫指定下列指get-deployment-instance令：

- 唯一部署 ID。若要取得部署識別碼，請呼叫清單部署命令。
- 唯一執行個體 ID。若要取得執行個體 ID，請呼叫指list-deployment-instances令。

若要檢視部署中使用之執行個體的 ID 清單，請呼叫命list-deployment-instances令，並指定：

- 唯一部署 ID。若要取得部署識別碼，請呼叫清單部署命令。
- 或者，是否僅由他們的部署狀態包含的特定執行個體 ID。(如果未指定，不論他們的部屬狀態如何，所有符合的執行個體都將列出)。

CodeDeploy 執行個體健

CodeDeploy 監控部署群組中執行個體的健全狀況狀態。如果正常運作的執行個體數目低於在部署期間針對部署群組所指定的最低正常運作執行個體數目，則它會讓部署失敗。例如，如果 85% 的執行個體在部署期間必須維持正常運作，而且部署群組包含 10 個執行個體，則部署至單一執行個體失敗時，整體部署就會失敗。原因是執行個體離線以安裝最新的應用程式修訂時，可用的正常運作執行個體計數已經降到 90%。失敗的執行個體加上另一個離線執行個體表示只有 80% 的執行個體運作良好且可用。CodeDeploy將失敗整體部署。

請務必記住，若要讓整體部署成功，必須符合下列條件：

- CodeDeploy 能夠部署到部署中的每個實例。
- 部署到至少一個執行個體必須成功。這表示即使最低正常運作主機值為 0，部署到至少一個執行個體還是必須成功 (也就是，至少有一個執行個體必須正常運作)，整體部署才會成功。

主題

運作狀態

CodeDeploy 為每個實例分配兩個健康狀態值：版本健康狀況和實例健康狀況。

修訂版運作狀態

修訂版運作狀況是根據目前在執行個體上安裝的應用程式修訂而定。它有下列的狀態數值：

- 目前：安裝在執行個體上的修訂版符合適用上一次成功部署中部署群組的修訂。
- 舊版：安裝在執行個體上的修訂版符合較舊版本的應用程式。
- 不明：應用程式修訂版尚未成功安裝在執行個體上。

執行個體運作狀態

執行個體運作狀態是根據成功部署至執行個體的而定。其具有下列數值：

- 正常運作：上一次部署到執行個體成功。
- 問題：嘗試部署修訂版到執行個體但失敗，或修訂版尚未部署到執行個體上。

CodeDeploy 使用修訂健康狀況和執行個體健康情況，以下列順序排程部署至部署群組的執行個體：

1. 有問題的執行個體運作狀態。
2. 不明修訂版的運作狀態。
3. 舊修訂版的運作狀態。
4. 目前修訂版的運作狀態。

如果整體部署成功，會更新修訂版和部署群組的運作狀態數值，以反映最新的部署狀況。

- 目前所有成功部署過的執行個體須保持為目前的狀態。否則，則會變成不名的狀態。
- 所有成功部署過的舊執行個體與不明的執行個體，須變為目前的狀態。否則，則會繼續維持舊版或不明的狀態。
- 目前所有成功部署過的正常運作執行個體，須保持正常運作狀態。否則，則會變成有問題狀態。
- 所有成功部署過的有問題之執行個體都會變為正常運作。否則，則會繼續保持為有問題狀態。

如果整體部署失敗或停止：

- CodeDeploy 嘗試部署應用程式修訂版本的每個執行個體都會將其執行個體健全狀況設定為狀況良好或狀況不良，這取決於該執行個體的部署嘗試是成功還是失敗。
- CodeDeploy 未嘗試部署應用程式修訂版本的每個執行個體，都會保留其目前的執行個體健全狀況值。
- 部署群組的修訂版保持不變。

關於健全狀態執行個體的最小數目

所需的最低正常運作執行個體數目定義為部署組態的一部分。

> **Important**
>
> 在藍/綠部署期間，部署組態和最低正常運作主機值會套用至替換環境中的執行個體，而不是原始環境中的執行個體。不過，從負載平衡器取消註冊原始環境中的執行個體時，如果無法成功取消註冊單一原始執行個體，則會將整體部署標示為 Failed (失敗)。

CodeDeploy 提供三種預設部署組態，這些組態具有常用的最低狀態主機值：

預設部署組態名稱	預先定義的最低正常運作主機值
CodeDeployDefault。 OneAtA 時代	1
CodeDeployDefault。 HalfAtA 時代	50%
CodeDeployDefault.AllAtOnce	0

如需預設部署組態的詳細資訊，請參閱使用中的部署組態 CodeDeploy。

您可以在中建立自訂部署組態， CodeDeploy 以定義自己的最低狀態主機值。在使用以下操作時，您可以自行定義數值 (整數或百分比皆可)：

- `minimum-healthy-hosts`當您使用中的create-deployment-config指令時一樣 AWS CLI。
- 就像 CodeDeploy API `Value` 中的MinimumHealthyHosts資料類型一樣。
- 就像您`MinimumHealthyHosts`在模板AWS::CodeDeploy::DeploymentConfig中使用時一 AWS CloudFormation 樣。

CodeDeploy 可讓您為兩個主要目的指定部署健康狀態良好的執行個體數目下限：

- 為判斷整體部署成功或失敗。如果應用程式修訂版成功部署最低數量的正常運作執行個體，則部署成功。
- 為判斷部署階段中，正常運作執行個體的數量，以允許部署繼續作業。

您可以為您的部署群組指定最低數量的正常運作執行個體，做為執行個體總數量的一部分或的百分比。如果您指定百分比，則在部署開始時，會將百分比 CodeDeploy 轉換為相等數目的執行個體，並將任何小數執行個體捨入。

CodeDeploy 在部署程序期間，追蹤部署群組執行個體的健全狀況狀態，並使用部署指定的運作狀態良好執行個體數目下限，判斷是否要繼續部署。基本原則是部署必須永遠不會使正常運作執行個體數量於您所指定的最小值。唯一例外是部署群組在起始時就擁有少於指定最低數量的執行個體。這種情況下，部署程序不會進一步減少任何正常運作執行個體。

> **Note**
>
> CodeDeploy 將嘗試部署到部署群組中的所有執行個體，甚至是目前處於 [已停止] 狀態的執行個體。在最低運作狀態良好的主機計算中，已停止的執行個體做為失敗的執行個體有相同的影響。若要解決因太多停止的執行個體導致部署失敗，請重新啟動執行個體，或變更其標籤，使其從部署群組中排除。

CodeDeploy 嘗試將應用程式修訂版部署到部署群組運作狀態不良的執行個體，以啟動部署程序。對於每次成功部署，請將執行個體的健全狀態 CodeDeploy 變更為狀況良好，並將其新增至部署群組的運作狀態良好的執行個 CodeDeploy 然後將目前狀態良好的執行個體數目與指定的運作狀態良好的執行個體數目下限

- 如果運作良好的執行個體數目小於或等於指定的運作狀態良好的執行個體數目下限，請 CodeDeploy 取消部署，以確保運作狀態良好的執行個體數量不會隨著部署次數而減少。
- 如果運作良好的執行個體數目超過指定的運作狀態良好執行個體數目下限至少一個，請將應用程式修訂版本 CodeDeploy 部署至原始健康狀態良好的執行個體集。

如果運作狀態良好的執行個體部署失敗，則會將該執行個體的健全狀態 CodeDeploy 變更為狀態不良。部署進行時，會 CodeDeploy 更新目前狀態良好的執行個體數目，並將其與指定的運作狀態良好的執行個體數目下限進行比較。如果在部署程序的任何時候，運作狀態良好的執行個體數目降至指定的最小數目，請 CodeDeploy 停止部署。此舉可防止接下來可能會失敗的部署，以及防止數量低於指定最小值的正常運作執行個體發生。

> **Note**
>
> 請確定您指定的正常運作執行個體最小值低於部署群組中執行個體的總數。若您指定的是百分比，請記住系統就自動將數值進位。否則，當部署開始時，運作狀態良好的執行個體數目已經小於或等於指定的運作狀態良好執行個體數目下限，而且整體部署 CodeDeploy 會立即失敗。

CodeDeploy 也會使用指定的運作狀態良好的執行個體數目下限，以及實際運作狀態良好的執行個體數目，來決定是否以及如何將應用程式修訂部署到多 根據預設，會將應用程式修訂版 CodeDeploy 部署到盡可能多的執行個體，而不會有運作狀態良好的執行個體數量低於指定的運作狀態良好執行個體數目下限的風險。

若要決定一次部署的執行個體數目，請 CodeDeploy 使用下列計算：

```
[total-hosts] - [minimum-healthy-hosts] =
        [number-of-hosts-to-deploy-to-at-once]
```

例如：

- 如果您的部署群組有 10 個執行個體，且您將運作狀態良好的執行個體數目下限設定為 9，則一次部 CodeDeploy 署至 1 個執行個體。
- 如果您的部署群組有 10 個執行個體，且您將運作狀態良好的執行個體數目下限設定為 3，請在第一個批次中同時部 CodeDeploy 署至 7 個執行個體，然後部署至第二個批次中剩餘的 3 個執行個體。
- 如果您的部署群組有 10 個執行個體，且您將運作狀態良好的執行個體數目下限設定為 0，請同時部 CodeDeploy 署至 10 個執行個體。

範例

以下範例假設部署群組有 10 個執行個體。

正常運作執行個體的最小值：95%

CodeDeploy 將運作良好的執行個體數目下限四捨五入至 10 個執行個體，這等於運作良好的執行個體數 在尚未部署修訂版至任何執行個體上，整體部署已立刻失敗。

正常運作執行個體的最小值：9

CodeDeploy 一次將修訂部署到一個執行個體。如果部署到任何執行個體失敗，則整體部署會 CodeDeploy 立即失敗，因為運作狀態良好的執行個體數目等於運作狀態良好的執行個體數目 此規則的例外是若最後一個執行個體發生失敗，仍然可以成功部署。

CodeDeploy 一次一個執行個體繼續部署，直到任何部署失敗或整體部署完成為止。如果所有 10 個部署皆成功，部署群組則會有 10 個正常運作執行個體。

正常運作執行個體的最小值：8

CodeDeploy 一次將修訂部署到兩個執行個體。如果其中兩個部署失敗，則整體部署會 CodeDeploy 立即失敗。此規則的例外是若最後一個執行個體是第二個發生失敗的，則仍然可以成功部署。

正常運作執行個體的最小值：0

CodeDeploy 一次將修訂部署到整個部署群組。至少必須有一個部署到執行個體成功，則整體部署才會成功。如果 0 個執行個體正常運作，則部署會失敗。這是因為要將整體部署標示為成功，整體部署完成時，至少一個執行個體必須為健康狀態良好，即使運作狀態最低的執行個體值為 0 也是如此。

關於每個可用區域的運作狀態良好的執行個體數

> **Note**
>
> 本節使用執行個體和主機可互換的術語來指向 Amazon EC2 執行個體。

如果您要部署到多個可用區域中的執行個體，您可以選擇性地啟用該zonal configuration功能，CodeDeploy 以便一次部署到一個可用區域。

啟用此功能時， CodeDeploy 將確保運作良好的主機數目維持在「每個區域運作狀態最低的主機」和「運作狀態最低主機」值之上。如果運作良好的主機數目低於任一值，則跨所有可用區域的部署 CodeDeploy 失敗。

若要計算一次部署的主機數目，請同時 CodeDeploy 使用「每個區域的運作狀況最低主機」和「運作狀態最低主機」值。 CodeDeploy 將使用較小的計算 [A][B]，其中[A]和[B]：

```
[A] = [total-hosts] - [min-healthy-hosts] =
        [number-of-hosts-to-deploy-to-at-once]
```

```
[B] = [total-hosts-per-AZ] - [min-healthy-hosts-per-AZ] =
        [number-of-hosts-to-deploy-to-at-once-per-AZ]
```

在確定要一次部署的主機數目之後，請按批次部 CodeDeploy 署到該數目的主機，一次部署一個可用區域，並在區域之間進行選擇性暫停 (或「烘烤時間」)。

範例

如果您的部署設定如下：

- [total-hosts] 是 200
- [minimum-healthy-hosts] 是 160
- [total-hosts-per-AZ] 是 100
- [minimum-healthy-hosts-per-AZ] 是 50

Then...

- [A] = 200 - 160 = 40
- [B] = 100 - 50 = 50
- 40小於 50

因此， CodeDeploy 將立即部署到40主機。

在這個案例中，部署會展開如下：

1. CodeDeploy 部署至第一個可用區域：
   a. CodeDeploy 部署到第一個40主機。

b. CodeDeploy 部署到下一個40主機。

c. CodeDeploy 部署到剩餘的20主機。

第一個可用區域的部署現在已完成。

2. (選擇性) 依監控持續時間或第一個區域設定新增監視持續時間所定義， CodeDeploy 等待部署至第一個區域「烘焙」。如果沒有問題，請 CodeDeploy 繼續。
3. CodeDeploy 部署至第二個可用區域：

a. CodeDeploy 部署到第一個40主機。

b. CodeDeploy 部署到下一個40主機。

c. CodeDeploy 部署到剩餘的20主機。

第二個也是最後一個可用區域的部署現已完成。

若要瞭解區域組態功能，以及如何指定每個可用區域的運作狀態良好的執行個體數目下限，請參閱zonal configuration。

使用中的部署組態 CodeDeploy

部署組態是一組規則，以及部署 CodeDeploy 期間使用的成功和失敗條件。這些規則和條件會有所不同，具體取決於您是部署到 EC2/現場部署運算平台、 AWS Lambda 運算平台還是 Amazon ECS 運算平台。

EC2/內部部署計算平台上的部署組態

當您部署到 EC2 /內部部署計算平台時，部署組態會透過使用「運作狀態最小的主機」值和選用的「每區域運作狀態最低主機」值，指定部署期間必須隨時保持可用的執行個體數目或百分比。

您可以使用由提供的三種預先定義的部署規劃之一， AWS 或建立自訂部署規劃。如需建立自訂部署規劃的詳細資訊，請參閱〈〉Create a Deployment Configuration。如果您未指定部署規劃，請 CodeDeploy 使用 CodeDeployDefault. OneAtA 時間部署配置。

如需部署期間如何 CodeDeploy 監視和評估執行個體健康狀態的詳細資訊，請參閱Instance Health。若要檢視已註冊至您 AWS 帳戶的部署設定清單，請參閱View Deployment Configuration Details。

EC2/內部部署計算平台的預定義部署組態

下表列出預先定義部署組態。

> **Note**
>
> 沒有支援此功能的預先定義部署組態 (此zonal configuration功能可讓您指定每個可用區域的運作狀況良好的主機數目)。如果要使用此功能，則必須建立自己的部署規劃。

部署組態	描述
CodeDeployDefault.AllAtOnce	就地部署： 嘗試部署應用程式修訂版，一次盡可能部署任意數量的執行個體。如果應用程式版本被部署到一個或多個執行個體，整體的部署狀態會顯示為成功。如果應用程式修訂版沒有部署到任何一個執行個體，整體的部署狀態會顯示為失敗。使用九個例證的範例 CodeDeployDefault。 AllAtOnce

部署組態	描述
	嘗試一次部署到所有九個執行個體。即使只有單一執行個體部署成功，整體部署也算成功。只有在部署到全部九個執行個體都失敗時，才算失敗。 藍/綠部署： • 部署至替代環境：遵循與相同的部署規則 CodeDeployDefault。 AllAtOnce 用於就地部署。 • 流量重新路由：立即路由流量到所有替換環境中的執行個體。如果流量成功重新路由到至少一個執行個體，便代表成功。重新路由到所有執行個體失敗之後，便是失敗。

部署組態	描述
CodeDeployDefault。 HalfAtaTime	就地部署： 一次部署到最多執行個體的一半 (分數無條件捨去)。如果應用程式修訂版部署到至少一半的執行個體，整體部署即為成功 (分數無條件計入)。否則，部署失敗。在九個執行個體的範例中，它將同時部署到最多四個執行個體。如果成功部署至五個或以上的執行個體，整體部署即為成功。否則，部署失敗。 **Note** 如果您要部署到多個 Auto Scaling 群組中的執行個體， CodeDeploy 則無論執行個體所在的 Auto Scaling 群組為何，一次最多可部署一半的執行個體。例如，假設您有兩個 Auto Scaling 群組，ASG1而且ASG2每個群組都有 10 個執行個體。在這個案例中，CodeDeploy 可能只部署到 10 個執行個體，ASG1並認為這是成功的，因為它已部署至少一半的執行個體。 藍/綠部署： • 部署至替代環境：遵循與相同的部署規則 CodeDeployDefault。 HalfAt適用於就地部署的 ATime。 • 流量重新路由：在替換環境中，流量最多一次路由到一半的執行個體。如果成功地重新路由到至少一半的執行個體，便是成功。否則，便會失敗。

部署組態	描述
CodeDeployDefault。 OneAtaTime	就地部署： 將應用程式修訂版一次部署到一個執行個體。 對於包含多個執行個體的部署群組： • 如果應用程式修訂版部署到所有執行個體，整體部署便算成功。此規則的例外是，若部署到最後一個執行個失敗，則整體部署仍然成功。這是因為一次只 CodeDeploy 允許一個執行個體離線 CodeDeployDefault。 OneAtA 時間配置。 • 一旦應用程式修訂版無法部署到最後一個執行個體以外的任何執行個，整體部署便告失敗。 • 在使用九個執行個體的範例中，它將一次部署到一個執行個體。如果部署到前八個執行個體成功，則整體部署成功。如果部署到前八個執行個體的任何一個失敗，則整體部署失敗。 對於僅包含一個執行個體的部署群組，如果部署到單一執行個體成功，整體部署便算成功。 藍/綠部署： • 部署至替代環境：遵循與相同的部署規則 CodeDeployDefault。 OneAt適用於就地部署的 ATime。 • 流量重新路由：在替換環境中，將一個執行個體的流量一次路由到一個執行個體。如果流量成功重新路由到所有的替代環境，便是成功。第一個路由失敗後，代表失敗。此規則的例外是，若最後一個執行個體無法註冊，則整體部署仍然成功。

在 Amazon ECS 運算平台上的部署組態

當您部署到 Amazon ECS 運算平台時，部署組態會指定流量如何轉移到更新的 Amazon ECS 任務集。您可以使用初期測試、線性或all-at-once部署組態來轉移流量。如需詳細資訊，請參閱 部署組態。

您也可以建立您自己的自訂 Canary 或線性部署組態。如需詳細資訊，請參閱 Create a Deployment Configuration。

適用於 Amazon ECS 運算平台的預先定義部署組態

下表列出可用於 Amazon ECS 部署的預先定義組態。

> **Note**
>
> 如果您使用的是 Network Load Balancer，則僅支援`CodeDeployDefault.ECSAllAtOnce`預先定義的部署設定。

部署組態	描述
CodeDeployDefault. 線性 PercentEvery 10 1 分鐘	每分鐘移動 10% 的流量，直到所有流量移動完畢。
CodeDeployDefault. 線上 10 3 分鐘 PercentEvery	每三分鐘移動 10% 的流量，直到所有流量移動完畢。
CodeDeployDefault. 百分比 10 分鐘	在第一個遞增中移動 10% 的流量。剩餘的 90% 會在五分鐘之後部署。
CodeDeployDefault. 百分之十 15 分鐘	在第一個遞增中移動 10% 的流量。剩餘的 90% 會在 15 分鐘之後部署。
CodeDeployDefault. ECS AllAtOnce	一次將所有流量轉移到更新後的 Amazon ECS 容器。

AWS CloudFormation 藍/綠部署的部署組態 (Amazon ECS)

當您透過 AWS CloudFormation 藍/綠部署部署到 Amazon ECS 運算平台時，部署組態會指定流量轉移到更新後的 Amazon ECS 容器的方式。您可以使用初期測試、線性或all-at-once部署組態來轉移流量。如需詳細資訊，請參閱 部署組態。

使用 AWS CloudFormation 藍/綠部署時，您無法建立自己的自訂初期測試或線性部署規劃。如需使用 AWS CloudFormation 管理 Amazon ECS 藍/綠部署的相關 step-by-step 說明，請參閱使用者指南 AWS CloudFormation中的 CodeDeploy 使用自動化 ECS 藍/綠部署。AWS CloudFormation

> **Note**
>
> 歐洲 (米蘭)、非洲 (開普敦) 和亞太區域 (大阪) 區域不提供 Amazon ECS 藍/綠部署的管理。AWS CloudFormation

AWS Lambda 運算平台上的部署組態

當您部署至 AWS Lambda 運算平台時，部署組態會指定流量轉移至應用程式中新 Lambda 函數版本的方式。您可以使用初期測試、線性或all-at-once部署組態來轉移流量。如需詳細資訊，請參閱 部署組態。

您也可以建立您自己的自訂 Canary 或線性部署組態。如需詳細資訊，請參閱 Create a Deployment Configuration。

AWS Lambda 運算平台的預先定義部署組態

下表列出 AWS Lambda 部署可用的預先定義組態。

部署組態	描述
CodeDeployDefault。 LambdaCanary10 百分比 5 分鐘	在第一個遞增中移動 10% 的流量。剩餘的 90% 會在五分鐘之後部署。
CodeDeployDefault。 LambdaCanary百分比 10 分鐘	在第一個遞增中移動 10% 的流量。剩餘的 90% 會在 10 分鐘之後部署。
CodeDeployDefault。 LambdaCanary百分之十 15 分鐘	在第一個遞增中移動 10% 的流量。剩餘的 90% 會在 15 分鐘之後部署。

部署組態	描述
CodeDeployDefault。 LambdaCanary百分之十 30 分鐘	在第一個遞增中移動 10% 的流量。剩餘的 90% 會在 30 分鐘之後部署。
CodeDeployDefault。 LambdaLinear十一Perc entEvery分鐘	每分鐘移動 10% 的流量，直到所有流量移動完畢。
CodeDeployDefault。 LambdaLinear十Perce ntEvery二分鐘	每兩分鐘移動 10% 的流量，直到所有流量移動完畢。
CodeDeployDefault。 LambdaLinear十Perce ntEvery三分鐘	每三分鐘移動 10% 的流量，直到所有流量移動完畢。
CodeDeployDefault。 LambdaLinearPercen tEvery十分鐘	每 10 分鐘移動 10% 的流量，直到所有流量移動完畢。
CodeDeployDefault.LambdaAllAtOnce	一次將所有流量轉移到更新的 Lambda 函數。

主題

- Create a Deployment Configuration
- View Deployment Configuration Details
- Delete a Deployment Configuration

使用建立部署規劃 CodeDeploy

如果您不想使用隨附的其中一個預設部署設定 CodeDeploy，您可以使用下列指示建立自己的部署設定。

您可以使用 CodeDeploy 主控台 AWS CLI、 CodeDeploy API 或 AWS CloudFormation 範本來建立自訂部署組態。

若要取得有關使用 AWS CloudFormation 樣板建立部署規劃的資訊，請參閱〈〉AWS CloudFormation CodeDeploy供參考的範本。

主題

- 建立部署規劃 (主控台)
- 使用 CodeDeploy (AWS CLI) 建立部署組態

建立部署規劃 (主控台)

使用下列指示，使用 AWS 主控台建立部署規劃。

CodeDeploy 使用主控台建立部署規劃

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在瀏覽窗格中，選擇 [部署規劃]。

   內建部署組態清單隨即顯示。

3. 選擇 Create deployment configuration (建立部署組態)。
4. 在部署規劃名稱中，輸入部署規劃的名稱。例如 **`my-deployment-config`**。
5. 在「運算平台」下，選擇下列其中一項：

   - EC2/內部部署
   - AWS Lambda
   - Amazon ECS

6. 執行以下任意一項：

   - 如果您選擇 EC2/ 內部部署：

     1. 在 [運作狀況良好的主機下限] 底下，指定部署期間必須隨時保持可用的執行個體數目或百分比。如需部署期間如何 CodeDeploy 監視和評估執行個體健康狀態的詳細資訊，請參閱Instance Health。
     2. (選擇性) 在區域組態下，選取啟用區域組態，讓應用程式一次 CodeDeploy 部署到一個區域內的一個 AWS 可用區域。透過一次部署到一個可用區域，您可以隨著對部署效能和可行性的信心提高，將部署公開給越來越大的受眾。如果您未啟用區域設定，請將應用程式 CodeDeploy 部署到跨區域隨機選取的主機。

如果您啟用區域組態功能，請注意下列事項：

- 只有在 Amazon EC2 執行個體的就地部署時才支援區域組態功能。(不支援藍/綠部署和內部部署執行個體)。如需就地部署的更多資訊，請參閱部署類型。
- 預先定義的部署規劃不支援區域組態功能。若要使用區域規劃，您必須建立自訂部署規劃，如此處所述。
- 如果 CodeDeploy 需要復原部署， CodeDeploy 將會在隨機主機上執行復原作業。（不CodeDeploy會一次回滾一個區域，如您所期望的那樣。）此復原行為是基於效能原因而選擇的。如需復原的詳細資訊，請參閱使用以下方式重新部署和復原部署 CodeDeploy。

3. 如果您選取了啟用區域組態核取方塊，請選擇性地指定下列選項：
   - (選擇性) 在監視持續時間中，指定完成可用區域部署後 CodeDeploy 必須等待的時間期間 (以秒為單位)。 CodeDeploy 在開始部署到下一個可用區域之前，會等待這段時間。請考慮新增監視器持續時間，讓部署有一些時間在一個可用區域中證明自己 (或「烘焙」)，然後再釋出下一個區域。如果您未指定監視持續時間，請立即 CodeDeploy 開始部署到下一個可用區域。如需監視持續時間設定如何運作的詳細資訊，請參閱關於每個可用區域的運作狀態良好的執行個體數。
   - (選擇性) 選取新增第一個區域的監視持續時間，以設定僅適用於第一個可用區域的監視持續時間。如果您想為第一個可用區域預留額外烘烤時間，可以設定此選項。如果您未在新增第一個區域監視器持續時間中指定值，則 CodeDeploy 會使用第一個可用區域的監視持續時間值。
   - (選擇性) 在每個區域運作良好的主機下限下，指定部署期間每個可用區域必須保持可用的執行個體數目或百分比。選擇「彈性 _ 百分比」以指定百分比，或選擇「主機 _ 計數」以指定數字。此欄位與健全狀況最低主機欄位搭配使用。如需詳細資訊，請參閱 關於每個可用區域的運作狀態良好的執行個體數。

     如果您未在每個區域的運作狀況良好的主機下限下指定值，則 CodeDeploy 會使用預設值0百分比。

- 如果您選擇AWS Lambda或 Amazon ECS：
  1. 對於「文字」，請選擇「線性」或「加
  2. 在「步驟」和「間隔」欄位中，執行下列其中一項作業：
     - 如果您選擇 Canary，在「步驟」中，輸入要轉移的流量百分比，介於 1 到 99 之間。這是第一個增量中移動的流量百分比。剩餘的流量會在第二個增量之選定間隔後轉移。

       在「間隔」中，輸入第一個和第二個流量偏移之間的分鐘數。

- 如果您選擇「線性」，對於「步驟」，請輸入要轉移的流量百分比，介於 1 到 99 之間。這是在每個間隔開始時轉移的流量百分比。

  在「間隔」中，輸入每個增量工作班次之間的分鐘數。

7. 選擇 Create deployment configuration (建立部署組態)。

   您現在擁有可與部署群組建立關聯的部署組態。

使用 CodeDeploy (AWS CLI) 建立部署組態

若要使用 AWS CLI 建立部署規劃，請呼叫指create-deployment-config令。

下列範例會建立名為的 EC2 /內部部署組態，ThreeQuartersHealthy該組態需要 75% 的目標執行個體在部署期間保持健康狀態：

```
aws deploy create-deployment-config --deployment-config-name ThreeQuartersHealthy --
minimum-healthy-hosts type=FLEET_PERCENT,value=75
```

下列範例會建立名為的 EC2 /內部部署組態，每個部署需300Total50PerAZ要 300 個目標執行個體維持健康狀態，而每個可用區域需要 50 個目標執行個體保持運作良好狀態。它還將監視器持續時間設置為 1 小時。

```
aws deploy create-deployment-config --deployment-config-name 300Total50PerAZ
 --minimum-healthy-hosts type=HOST_COUNT,value=300 --zonal-config
 '{"monitorDurationInSeconds":3600,"minimumHealthyHostsPerZone":
{"type":"HOST_COUNT","value":50}}'
```

下列範例會建立名為的 AWS Lambda 部署組態Canary25Percent45Minutes。它使用 Canary 轉換功能，轉換第一個遞增的 25% 的流量。剩餘的 75% 會在 45 分鐘之後轉移。

```
aws deploy create-deployment-config --deployment-config-name Canary25Percent45Minutes
 --traffic-routing-config
 "type="TimeBasedCanary",timeBasedCanary={canaryPercentage=25,canaryInterval=45}" --
compute-platform Lambda
```

下列範例會建立名為的 Amazon ECS 部署組Canary25Percent45Minutes態。它使用 Canary 轉換功能，轉換第一個遞增的 25% 的流量。剩餘的 75% 會在 45 分鐘之後轉移。

```
aws deploy create-deployment-config --deployment-config-name Canary25Percent45Minutes
 --traffic-routing-config
 "type="TimeBasedCanary",timeBasedCanary={canaryPercentage=25,canaryInterval=45}" --
compute-platform ECS
```

檢視部署組態詳細資料 CodeDeploy

您可以使用主 CodeDeploy 控台 AWS CLI、或 CodeDeploy API 來檢視與您 AWS 帳戶相關聯之部署組態的詳細資料。如需預先定義部 CodeDeploy 署規劃的描述，請參閱EC2/內部部署計算平台的預定義部署組態。

主題

- 檢視部署組態詳細資料 (主控台)
- 檢視部署組態 (CLI)

檢視部署組態詳細資料 (主控台)

若要使用 CodeDeploy 主控台檢視部署規劃名稱清單：

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在瀏覽窗格中，展開 [部署]，然後選擇 [部署設定]。

   您可以在此看到部署組態名稱和每個部署組態的條件。

   > **Note**
   >
   > 如果未顯示任何項目，請確定已選取正確的區域。在導覽列的區域選取器中，選擇「區域」和「端點」中列出的其中一個區域AWS 一般參考。 CodeDeploy 僅在這些地區支援。

檢視部署組態 (CLI)

若要使用 AWS CLI 檢視部署組態詳細資料，請呼叫`get-deployment-config`命令或命`list-deployment-configs`令。

若要檢視有關單一部署規劃的詳細資料，請呼叫指get-deployment-config令，指定唯一的部署規劃名稱。

若要檢視有關多個部署組態的詳細資料，請呼叫清單部署指令。

使用刪除部署規劃 CodeDeploy

您可以使用 AWS CLI 或 CodeDeploy API 刪除與您 AWS 帳戶相關聯的自訂部署組態。您無法刪除內建的部署組態，例如 `CodeDeployDefault.AllAtOnce`、`CodeDeployDefault.HalfAtATime` 和 `CodeDeployDefault.OneAtATime`。

> **Warning**
>
> 您無法刪除仍在使用中的自訂部署組態。如果您刪除一個未使用的自訂部署組態，您將無法再為它與新的部署及新的部署群組建立關聯性。這個操作無法復原。

若要使用 AWS CLI 刪除部署規劃，請呼叫指delete-deployment-config令，指定部署規劃名稱。若要檢視部署規劃名稱清單，請呼叫指list-deployment-configs令。

下列範例會刪除名為的部署規劃 ThreeQuartersHealthy。

```
aws deploy delete-deployment-config --deployment-config-name ThreeQuartersHealthy
```

使用中的應用程式 CodeDeploy

設定執行個體之後，但您必須在中建立應用程式，才能部署修訂版本 CodeDeploy。應用程式只是一個名稱或容器，用 CodeDeploy 來確保在部署期間參考正確的修訂版本、部署組態和部署群組。

使用下表中的資訊中的後續步驟：

運算平台	案例	後續步驟的資訊
EC2/內部部署	我尚未建立執行個體。	請參閱使用的例證 CodeDeploy，然後返回此頁面。
EC2/內部部署	我已建立執行個體，但沒有完成標記。	請參閱Tagging Instances for Deployments，然後返回此頁面。
EC2/ 現場部署、AWS Lambda 和 Amazon ECS	我尚未建立應用程式。	請參閱建立應用程式 CodeDeploy
EC2/ 現場部署、AWS Lambda 和 Amazon ECS	我已經建立一個應用程式，但尚未建立部署群組。	請參閱建立部署群組 CodeDeploy。
EC2/ 現場部署、AWS Lambda 和 Amazon ECS	我已經建立一個應用程式及部署群組，但尚未建立應用程式修訂版。	請參閱使用的應用程式修訂 CodeDeploy。
EC2/ 現場部署、AWS Lambda 和 Amazon ECS	我已經建立一個應用程式及部署群組，並且已經上傳我的應用程式修訂版。我可以開始部署。	請參閱使用建立部署 CodeDeploy。

主題

- 建立應用程式 CodeDeploy
- 檢視申請詳細資料 CodeDeploy
- 建立通知規則
- 重命名 CodeDeploy 應用程式

- 在中刪除應用程式 CodeDeploy

建立應用程式 CodeDeploy

應用程式只是用 CodeDeploy 來確保在部署期間參考正確的修訂版本、部署組態和部署群組的名稱或容器。您可以使用 CodeDeploy 主控台 AWS CLI、 CodeDeploy API 或 AWS CloudFormation 範本來建立應用程式。

您的程式碼或應用程式修訂版會透過稱為部署的程序，安裝至執行個體。 CodeDeploy 支援兩種部署類型：

- 就地部署：停止部署群組中每個執行個體上的應用程式、安裝最新的應用程式修訂版本，並啟動和驗證新版本的應用程式。您可以使用負載平衡器，以便在部署期間取消註冊每個執行個體，然後在部署完成後還原至服務。只有使用 EC2 /內部部署計算平台的部署才能使用就地部署。如需就地部署的更多資訊，請參閱就地部署概述。
- 藍/綠部署：部署的行為取決於您使用的運算平台：
  - EC2 /內部部署計算平台上的藍色/綠色：部署群組 (原始環境) 中的執行個體會使用下列步驟取代為不同的一組執行個體 (取代環境)：
    - 針對替代環境佈建執行個體。
    - 最新的應用程式修訂版會安裝在取代執行個體上。
    - 應用程式測試和系統驗證等活動會發生選擇性的等待時間。
    - 取代環境中的執行個體會使用一或多個 Elastic Load Balancing 負載平衡器登錄，導致流量重新路由傳送到這些執行個體。原始環境中的執行個體會取消註冊，並可終止或繼續執行以供其他用途使用。

    > **Note**
    >
    > 如果您使用 EC2 /內部部署運算平台，請注意藍/綠部署僅適用於 Amazon EC2 執行個體。

  - AWS Lambda 或 Amazon ECS 運算平台上的藍色/綠色：流量會根據初期測試、線性或all-at-once部署組態以遞增方式移動。
  - 透過藍/綠部署 AWS CloudFormation：流量會從您目前的資源轉移到更新的資源，做為 AWS CloudFormation 堆疊更新的一部分。目前僅支援 ECS 藍/綠部署。

  如需藍/綠部署的詳細資訊，請參閱 藍/綠部署概述。

當您使用 CodeDeploy 主控台建立應用程式時，您可以同時設定其第一個部署群組。當您使用建立應 AWS CLI 用程式時，請在單獨的步驟中建立其第一個部署群組。

若要檢視已在您 AWS 帳戶中註冊的應用程式清單，請參閱檢視申請詳細資料 CodeDeploy。如需有關使用 AWS CloudFormation 範本建立應用程式的資訊，請參閱AWS CloudFormation CodeDeploy供參考的範本。

這兩種部署類型不適用於所有目的地。下表列出哪些部署類型可用於部署至三種類型的部署目的地。

部署目的地	就地	藍/綠
Amazon EC2	是	是
現場部署	是	否
無伺服器 AWS Lambda 函數	否	是
Amazon ECS 應用	否	是

主題

建立就地部署的應用程式 (主控台)

若要使用 CodeDeploy 主控台為就地部署建立應用程式，請執行下列動作：

> **Warning**
>
> 如果發生下列情況，請勿採用這些步驟：
>
> - 您尚未準備好要用於 CodeDeploy 部署的執行個體。若要設定您的執行個體，請遵循使用的例證 CodeDeploy中的說明，然後遵循本主題中的步驟。

- 您希望建立使用自訂部署組態的應用程式，但您尚未建立部署組態。請遵循Create a Deployment Configuration中的說明，再返回本主題中的步驟。
- 您沒有信任 CodeDeploy 具有最低必要信任和權限的服務角色。若要使用必要的許可建立及設定服務角色，請遵循步驟 2：建立服務角色 CodeDeploy中的說明，再返回本主題中的步驟。
- 您想要在 Elastic Load Balancing 中為就地部署選取 Classic Load Balancer、應用程式負載平衡器或 Network Load Balancer，但尚未建立它。

若要使用主控台為就地部署建立應用程 CodeDeploy 式：

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在導覽窗格中，展開 Deploy (部署)，然後選擇 Getting started (入門)。
3. 選擇建立應用程式。
4. Application name (應用程式名稱) 中輸入您應用程式的名稱。
5. 在 Compute Platform (運算平台) 中，選擇 EC2/On-premises (EC2/現場部署)。
6. 選擇建立應用程式。
7. 在您的應用程式頁面，從 Deployment groups (部署群組) 標籤中，選擇 Create deployment group (建立部署群組)。
8. 在 Deployment group name (部署群組名稱) 中，輸入描述部署群組的名稱。

   > **Note**
   >
   > 如果您想要使用在其他部署群組中使用的相同設定 (包括部署群組名稱、標籤、Amazon EC2 Auto Scaling 群組名稱或兩者；以及部署組態)，請在此頁面上指定這些設定。雖然這個新的部署群組和現有的部署群組具有相同的名稱，但會 CodeDeploy 將它們視為個別的部署群組，因為它們都與不同的應用程式相關聯。

9. 在服務角色中，選擇授與目標執行個體 CodeDeploy存取權的服務角色。
10. 在 Deployment type (部署類型) 中，選擇 In-place (就地)。

11. 在 Environment configuration (環境資訊) 中，選取下列任何項目：

    a. Amazon EC2 Auto Scaling 群組：輸入或選擇要將應用程式修訂部署到的 Amazon EC2 Auto Scaling 群組的名稱。當新的 Amazon EC2 執行個體作為 Amazon EC2 自動擴展群組的一部分啟動時， CodeDeploy 可以自動將您的修訂部署到新執行個體。您最多可以將 10 個 Amazon EC2 Auto Scaling 群組新增至一個部署群組。

    b. Amazon EC2 執行個體或現場部署執行個體：在「金鑰」和「值」欄位中，輸入用於標記執行個體的金鑰值組的值。您最多可以在單一標籤群組內標記 10 個金鑰值對。

       i. 您可以在「值」欄位中使用萬用字元來識別以特定模式標記的所有執行個體，例如類似的 Amazon EC2 執行個體、成本中心和群組名稱等。例如，如果您在「關鍵字」欄位中選擇「名稱」並**GRP-*a**在「值」欄位中輸入，則會 CodeDeploy 識別符合該陣列的所有例證**GRP-1a**，例如**GRP-2a**、和**GRP-XYZ-a**。

       ii. Value (值) 欄位區分大小寫。

       iii. 若要從清單中移除一個金鑰值對，請選擇 Remove tag (移除標籤)。

       CodeDeploy 尋找符合每個指定索引鍵值對或 Amazon EC2 Auto Scaling 群組名稱的執行個體時，會顯示相符執行個體的數量。選擇數字即可查看有關於執行個體的詳細資訊。

       如果您想要強化部署到執行個體的條件，請選擇 Add tag group (新增標籤群組) 以建立標籤群組。您可以建立最多三個標籤群組，每個標籤群組最多包含 10 個金鑰值對。當您在部署群組中使用多個標籤群組，只有所有標籤群組識別出的執行個體會包含在部署群組中。這表示一個執行個體至少有一個標籤必須符合部署群組中的每個群組。

       如需使用標籤群組來強化部署群組的相關資訊，請參閱Tagging Instances for Deployments。

12. 在 Deployment settings (部署設定) 中，選擇部署組態以控制應用程式部署到執行個體的速度，例如一次一個或一次全部。如需部署組態的詳細資訊，請參閱使用中的部署組態 CodeDeploy。

13. (選擇性) 在負載平衡器中，選取啟用負載平衡，然後從清單中選取傳統負載平衡器、Application Load Balancer 目標群組和 Network Load Balancer 目標群組，以便在 CodeDeploy 部署期間管理執行個體的流量。您最多可以選取 10 個傳統負載平衡器和 10 個目標群組，總共 20 個項目。確保要部署到的 Amazon EC2 執行個體已向所選負載平衡器 (傳統負載平衡器) 或目標群組 (應用程式負載平衡器和網路負載平衡器) 註冊。

    在部署期間，原始執行個體會從選取的負載平衡器和目標群組取消註冊，以防止在部署期間將流量路由傳送到這些執行個體。部署完成後，每個執行個體都會重新註冊所有選取的傳統負載平衡器和目標群組。

如需 CodeDeploy 部署負載平衡器的詳細資訊，請參閱Integrating CodeDeploy with Elastic Load Balancing。

14. (選擇性) 展開進階，然後設定要包含在部署中的任何選項，例如 Amazon SNS 通知觸發器、Amazon CloudWatch 警示或自動復原。

    如需詳細資訊，請參閱 設定部署群組的進階選項。

15. 選擇 Create deployment group (建立部署群組)。

在下一個步驟中，您要準備一個修訂版本，以便將其部署至應用程式和部署群組。如需說明，請參閱使用的應用程式修訂 CodeDeploy。

建立藍色/綠色部署 (主控台) 的應用程式

若要使用 CodeDeploy 主控台建立藍/綠部署的應用程式：

> **Note**
>
> AWS Lambda 運算平台的部署永遠是藍/綠部署。您不指定部署類型選項。

> **Warning**
>
> 如果發生下列情況，請勿採用這些步驟：
>
> - 在藍/綠部署程序期間，您沒有安裝 CodeDeploy 代理程式的執行個體要取代。若要設定您的執行個體，請遵循使用的例證 CodeDeploy中的說明，然後遵循本主題中的步驟。
> - 您希望建立使用自訂部署組態的應用程式，但您尚未建立部署組態。請遵循Create a Deployment Configuration中的說明，再返回本主題中的步驟。
> - 您沒有至少信任信任中所述 CodeDeploy 的信任和權限的服務角色步驟 2：建立服務角色 CodeDeploy。若要建立及設定服務角色，請遵循步驟 2：建立服務角色 CodeDeploy中的說明，再返回本主題中的步驟。
> - 您尚未在 Elastic Load Balancing 中建立 Classic Load Balancer、應用程式負載平衡器或 Network Load Balancer 來註冊替代環境中的執行個體。如需詳細資訊，請參閱 在 Elastic Load Balancing 中為 CodeDeploy Amazon EC2 部署設定負載平衡器。

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在導覽窗格中，展開 Deploy (部署)，然後選擇 Getting started (入門)。
3. Application name (應用程式名稱) 中輸入您應用程式的名稱。
4. 在 Compute platform (運算平台) 中，選擇 EC2/On-Premises (EC2/現場部署)。
5. 選擇建立應用程式。
6. 在您的應用程式頁面，從 Deployment groups (部署群組) 標籤中，選擇 Create deployment group (建立部署群組)。
7. 在 Deployment group name (部署群組名稱) 中，輸入描述部署群組的名稱。

   > **Note**
   >
   > 如果您想要使用在其他部署群組中使用的相同設定 (包括部署群組名稱標籤、Amazon EC2 Auto Scaling 群組名稱和部署組態)，請在此頁面上選擇這些設定。雖然這個新的部署群組和現有的部署群組具有相同的名稱，但會 CodeDeploy 將它們視為個別的部署群組，因為每個部署群組都與個別的應用程式相關聯。

8. 在服務角色中，選擇授與目標執行個體 CodeDeploy存取權的服務角色。
9. 在 Deployment type (部署類型) 中，選擇 Blue/green (藍/綠)。
10. 在 Environment configuration (環境組態中) 中，選擇為您的替換環境提供執行個體的方法：

    a. 自動複製 Amazon EC2 自 Auto Scaling 群組： CodeDeploy 透過複製您指定的群組來建立 Amazon EC2 Auto Scaling 群組。

    b. 手動佈建執行個體：直到建立部署，您才能為您的替換環境指定執行個體。開始部署之前，您必須建立執行個體。在這個選項中，您要改為指定欲取代的執行個體。

11. 根據您在步驟 10 中所做的選擇，執行下列任一作業：

    - 如果您選擇「自動複製 Amazon EC2 Auto Scaling」群組：在 Amazon EC2 Auto Scaling 群組中，選擇或輸入您想要用作 Amazon EC2 Auto Scaling 群組範本的 Amazon EC2 Auto Scaling 群組名稱，以便替換環境中的執行個體使用。您選擇的 Amazon EC2 Auto Scaling 群組中目前運作良好的執行個體數目是在替代環境中建立的。

- 如果您選擇手動佈建執行個體：啟用 Amazon EC2 Auto Scaling 群組、Amazon EC2 執行個體或兩者，以指定要新增至此部署群組的執行個體。輸入 Amazon EC2 標籤值或 Amazon EC2 Auto Scaling 群組名稱，以識別原始環境中的執行個體 (也就是您要取代或執行目前應用程式修訂版的執行個體)。

12. 在負載平衡器中，選取啟用負載平衡，然後從清單中選取要向其註冊替代 Amazon EC2 執行個體的傳統負載平衡器、應用程式負載平衡器目標群組和 Network Load Balancer 目標群組。每個取代執行個體都會向所有選取的傳統負載平衡器和目標群組登錄。您最多可以選取 10 個傳統負載平衡器和 10 個目標群組，總共 20 個項目。

    系統會根據您選擇的流量重新路由傳送和部署組態設定，將流量從原始執行個體重新路由傳送至替代執行個體。

    如需 CodeDeploy 部署負載平衡器的詳細資訊，請參閱Integrating CodeDeploy with Elastic Load Balancing。

13. 在 Deployment settings (部署設定) 中，檢閱重新路由流量至替換環境的預設選項、要用於部署的部署組態，以及部署後處理原始環境中執行個體的方式。

    若您想要變更設定，請繼續下一個步驟。否則，請跳至步驟 15。

14. 若要變更藍/綠部署的部署設定，請變更下列任何設定。

設定	選項
Traffic rerouting (重新路由流量)	• 立即重新路由流量：一旦在取代環境中佈建執行個體並在其上安裝了最新的應用程式修訂版本，就會自動向指定的負載平衡器和目標群組登錄這些執行個體，進而導致流量重新路由傳送給這些執行個體。然後撤銷註冊原始環境中的執行個體。 • 我將選擇是否重新路由傳送流量：除非您手動重新路由傳送流量，否則取代環境中的執行個體不會向指定的負載平衡器和目標群組登錄。如果過了您指定的等待時間卻沒有重新路由流量，則部署狀態會變更為「已停止」。

設定	選項
Deployment configuration (部署組態)	選擇替代環境中的執行個體向負載平衡器和目標群組註冊的速率，例如一次登錄一個執行個體或一次全部登錄。 **Note** 流量成功路由到替換環境之後，無論選取哪一種部署設定，都會立即將原始環境中的執行個體全部撤銷註冊。 如需詳細資訊，請參閱 使用中的部署組態 CodeDeploy。
Original instances (原始執行個體)	• 終止部署群組中的原始執行個體：流量重新路由傳送至取代環境後，從負載平衡器和目標群組取消註冊的執行個體會在您指定的等待期間後終止。 • 讓部署群組中的原始執行個體保持執行中：將流量重新路由傳送至取代環境後，從負載平衡器和目標群組取消註冊的執行個體會保持在執行狀態。

15. (選擇性) 在進階中，設定要包含在部署中的選項，例如 Amazon SNS 通知觸發器、Amazon CloudWatch 警示或自動復原。

    如需在部署群組中指定進階選項的相關資訊，請參閱設定部署群組的進階選項。

16. 選擇 Create deployment group (建立部署群組)。

在下一個步驟中，您要準備一個修訂版本，以便將其部署至應用程式和部署群組。如需說明，請參閱使用的應用程式修訂 CodeDeploy。

為 Amazon ECS 服務部署建立應用程式 (主控台)

您可以使用主 CodeDeploy 控台為 Amazon ECS 服務部署建立應用程式。

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在瀏覽窗格中，展開 [部署]，然後選擇 [開始使用]。
3. 在 [建立應用程式] 頁面上選擇 [使用] CodeDeploy。
4. Application name (應用程式名稱) 中輸入您應用程式的名稱。
5. 在運算平台中，選擇 Amazon ECS。
6. 選擇建立應用程式。
7. 在您的應用程式頁面，從 Deployment groups (部署群組) 標籤中，選擇 Create deployment group (建立部署群組)。如需針對 Amazon ECS 部署建立部署群組所需項目的詳細資訊，請參閱在您開始 Amazon ECS 部署之前。
8. 在 Deployment group name (部署群組名稱) 中，輸入描述部署群組的名稱。

   > **Note**
   >
   > 如果您想使用用於其他部署群組的相同設定 (包括部署群組名稱和部署組態)，請在本頁面上選擇這些設定。雖然這個新群組和現有群組可能具有相同的名稱，但是會 CodeDeploy 將它們視為個別的部署群組，因為每個群組都與個別的應用程式相關聯。

9. 在服務角色中，選擇授與 Amazon ECS CodeDeploy 存取權的服務角色。如需詳細資訊，請參閱步驟 2：建立服務角色 CodeDeploy。
10. 從負載平衡器名稱中，選擇為 Amazon ECS 服務提供流量的負載平衡器名稱。
11. 從生產接聽程式連接埠中，選擇為向 Amazon ECS 服務提供生產流量的接聽程式的連接埠和通訊協定。
12. (選擇性) 從測試接聽程式連接埠中，選擇測試接聽程式的連接埠和通訊協定，以在部署期間為 Amazon ECS 服務中設定的替換任務提供流量。您可以在`AfterAllowTestTraffic`掛接期間執行的 AppSpec 檔案中指定一或多個 Lambda 函數。這些函數可以運行驗證測試。如果驗證測試失敗，則會觸發部署復原。如果驗證測試成功，則會觸發部署生命週期中的下一個勾點`BeforeAllowTraffic`。如果未指定測試接聽程式連接埠，則`AfterAllowTestTraffic`掛接期間不會發生任何事情。如需詳細資訊，請參閱 AppSpec Amazon ECS 部署的「掛鉤」部分。

13. 從目標群組 1 名稱和目標群組 2 名稱中，選擇部署期間用於路由傳送流量的目標群組。CodeDeploy 將一個目標群組繫結至 Amazon ECS 服務的原始任務集，另一個目標群組繫結至其替換任務集。如需詳細資訊，請參閱應用程式負載平衡器的目標群組。
14. 選擇「立即重新路由流量」或「指定何時重新路由流量」，以確定何時將流量重新路由至更新的 Amazon ECS 服務。

    如果您選擇「立即重新路由傳送流量」，則部署會在佈建取代工作集後自動重新路由傳送流量。

    如果您選擇指定重新路由傳送流量的時間，請選擇成功佈建取代工作集後要等待的天數、小時數和分鐘數。在此等待時間內，會執行 AppSpec 檔案中指定的 Lambda 函數中的驗證測試。如果等待時間在重新路由傳送流量之前過期，則部署狀態會變更為`Stopped`。
15. 對於原始修訂終止，請選擇在成功部署後要等待的天數、小時數和分鐘數，然後在 Amazon ECS 服務中設定的原始任務終止。
16. (選擇性) 在進階中，設定您要包含在部署中的任何選項，例如 Amazon SNS 通知觸發器、Amazon CloudWatch 警示或自動復原。

    如需詳細資訊，請參閱 設定部署群組的進階選項。

建立 AWS Lambda 函數部署的應用程式 (主控台)

您可以使用主 CodeDeploy 控台為 AWS Lambda 函數部署建立應用程式。

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在瀏覽窗格中，展開 [部署]，然後選擇 [開始使用]。
3. 在 [建立應用程式] 頁面上選擇 [使用] CodeDeploy。
4. 以 Application name (應用程式的名稱) 輸入您應用程式的名稱。
5. 從 Compute platform (運算平台)，選擇 AWS Lambda。
6. 選擇建立應用程式。
7. 在您的應用程式頁面，從 Deployment groups (部署群組) 標籤中，選擇 Create deployment group (建立部署群組)。

8. 在 Deployment group name (部署群組名稱) 中，輸入描述部署群組的名稱。

> **Note**
>
> 如果您想使用用於其他部署群組的相同設定 (包括部署群組名稱和部署組態)，請在本頁面上選擇這些設定。雖然這個新的部署群組和現有的部署群組可能具有相同的名稱，但會 CodeDeploy 將它們視為個別的部署群組，因為每個部署群組都與個別的應用程式相關聯。

9. 在服務角色中，選擇授與 CodeDeploy存取權的服務角色 AWS Lambda。如需詳細資訊，請參閱步驟 2：建立服務角色 CodeDeploy。

10. 如果您想要使用預先定義的部署組態，請從 Deployment configuration (部署組態) 中選擇一個，然後跳到步驟 12。若要建立自訂組態，請繼續下一個步驟。

    如需部署組態的詳細資訊，請參閱AWS Lambda 運算平台上的部署組態。

11. 若要建立自訂組態，請選擇 Create deployment configuration (建立部署組態)，然後執行下列動作：

    a. 對於 Deployment configuration name (部署組態名稱)，為組態輸入名稱。

    b. 在 Type (類型) 中，選擇組態類型。若選擇 Canary (Canary)，系統會在兩次遞增中逐漸轉移流量。若選擇 Linear (線性)，流量以相等增量轉移，每個增量之間分鐘數相等。

    c. 對於Step (步驟)，輸入要轉移的 1 到 99 之間的流量百分比。如果您的組態類型為 Canary，這是在第一個增量轉移的流量百分比。剩餘的流量會在第二個增量之選定間隔後轉移。如果組態類型為 Linear (線性)，則代表每個間隔開始時轉移的流量百分比。

    d. 在 Interval (間隔) 中，輸入分鐘數。如果組態類型是 Canary (Canary)，這就代表第一次和第二次流量轉移時間隔的分鐘數。如果您的組態類型是 Linear (線性)，這是每個增量轉移之間的分鐘數。

    > **Note**
    >
    > AWS Lambda 部署的長度上限為兩天或 2,880 分鐘。因此，Canary 組態的 Interval (間隔) 最大指定值為 2,800 分鐘。線性組態的最大值取決於 Step (步驟) 的值。例如，如果線性流量轉移的步驟百分比是 25%，則有四個流量移位。最大間隔值為 2,880 除以 4，或是 720 分鐘。

    e. 選擇 Create deployment configuration (建立部署組態)。

12. (選擇性) 在進階中，設定您要包含在部署中的任何選項，例如 Amazon SNS 通知觸發器、Amazon CloudWatch 警示或自動復原。

    如需詳細資訊，請參閱 設定部署群組的進階選項。

13. 選擇 Create deployment group (建立部署群組)。

建立應用程式 (CLI)

若要使用 AWS CLI 建立應用程式，請呼叫 create-Application 指令，指定唯一代表應用程式的名稱。（在一個 AWS 帳戶中，每個地區只能使用一次 CodeDeploy 應用程序名稱。 您可以在不同的區域重複使用應用程式名稱。）

使用建立應用 AWS CLI 程式之後，下一個步驟是建立部署群組，以指定要部署修訂版本的執行個體。如需說明，請參閱建立部署群組 CodeDeploy。

在您建立部署群組後，下一步是準備要部署到應用程式和部署群組的修訂版。如需說明，請參閱 使用的應用程式修訂 CodeDeploy。

檢視申請詳細資料 CodeDeploy

您可以使用主 CodeDeploy 控台 AWS CLI、或 CodeDeploy API 來檢視與您 AWS 帳戶相關聯之所有應用程式的詳細資料。

主題

- 檢視應用程式詳細資料 (主控
- 檢視應用程式詳細資料 (CLI)

檢視應用程式詳細資料 (主控

若要使用 CodeDeploy 主控台檢視應用程式詳細資訊：

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在瀏覽窗格中，展開 [部署]，然後選擇 [開始使用]。
3. 要查看其他應用程式的詳細資訊，請選擇清單中的應用程式名稱。

檢視應用程式詳細資料 (CLI)

若要使用 AWS CLI 檢視應用程式詳細資訊，請呼叫命batch-get-application令、指令或命list-applications令。get-application

若要檢視單一應用程式的詳細資訊，請呼叫 get-應用程式命令並指定應用程式名稱。

若要檢視多個應用程式的詳細資訊，請呼叫指batch-get-applications令，指定多個應用程式名稱。

若要檢視應用程式名稱清單，請呼叫清單應用程式命令。

建立通知規則

您可以使用通知規則，在部署應用程式發生變更時通知使用者，例如部署成功和部署失敗。通知規則指定用於傳送通知的事件和 Amazon SNS 主題。如需詳細資訊，請參閱什麼是通知？

您可以使用主控台或 AWS CLI 建立的通知規則 AWS CodeDeploy。

建立通知規則 (主控台)

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy/。
2. 選擇 Application (應用程式)，然後選擇要新增通知的應用程式。
3. 在應用程式頁面上，選擇 Notify (通知)，然後選擇 Create notification rule (建立通知規則)。您也可以移至應用程式的 Settings (設定) 頁面，然後選擇 Create notification rule (建立通知規則)。
4. 在 Notification name (通知名稱) 中，輸入規則的名稱。
5. 如果您只想要提供給 Amazon 的資訊 EventBridge 包含在通知中，請在 [詳細資料類型] 中選擇 [基本]。如果您想要包含提供給 Amazon 的資訊以 EventBridge 及可能由 CodeDeploy 或通知管理員提供的資訊，請選擇「完整」。

   如需詳細資訊，請參閱瞭解通知內容和安全性。
6. 在 Events that trigger notifications (觸發通知的事件) 中，選取您要傳送通知的事件。

類別	事件
部署	失敗 Succeeded 已開始

7. 在 Targets (目標) 中，選擇 Create SNS topic (建立 SNS 主題)。

> **Note**
>
> 當您建立主題時，會套用 CodeDeploy 允許將事件發佈至主題的原則。使用專門針對 CodeDeploy 通知建立的主題也有助於確保您只將使用者新增至您想要查看有關此部署應用程式之通知之該主題的訂閱清單。

在 codestar-notifications- 字首之後，輸入主題的名稱，然後選擇 Submit (提交)。

> **Note**
>
> 如果您要使用現有 Amazon SNS 主題而非建立新主題，請在 Targets (目標) 中選擇其 ARN。請確定主題具有適當的存取政策，而且訂閱者清單只包含允許查看部署應用程式相關資訊的使用者。如需詳細資訊，請參閱針對通知設定現有 Amazon SNS 主題和了解通知內容和安全性。

8. 若要完成建立規則，請選擇 Submit (提交)。
9. 您必須先訂閱使用者訂閱規則的 Amazon SNS 主題，才能收到通知。如需詳細資訊，請參閱訂閱使用者訂閱屬於目標的 Amazon SNS 主題。您也可以設定通知之間的整合，並 AWS Chatbot 將通知傳送到 Amazon Chime 聊天室或 Slack 頻道。如需詳細資訊，請參閱設定通知與之間的整合 AWS Chatbot。

建立通知規則 (AWS CLI)

1. 在終端機或命令提示字元中，執行 create-notification rule 命令以產生 JSON 架構：

```
aws codestar-notifications create-notification-rule --generate-cli-skeleton
 > rule.json
```

您可以將檔案命名為任何您想要的名稱。在此範例中，檔案命名為 *rule.json*。

2. 在純文字編輯器中開啟 JSON 檔案，然後對其進行編輯，以包含您要用於規則的資源、事件類型和 Amazon SNS 目標。下列範例會顯示 AWS 針對識別碼**MyNotificationRule**為 *123* 456789012 的帳戶*MyDeploymentApplication*中名為的應用程式命名的通知規則。MyNotificationTopic當部署成功時，會將通知與完整詳細資料類型一起傳送至名為 *codestar-##*的 Amazon SNS 主題：

```
{
    "Name": "MyNotificationRule",
    "EventTypeIds": [
        "codedeploy-application-deployment-succeeded"
    ],
    "Resource": "arn:aws:codebuild:us-east-2:123456789012:MyDeploymentApplication",
    "Targets": [
        {
            "TargetType": "SNS",
            "TargetAddress": "arn:aws:sns:us-east-2:123456789012:codestar-
notifications-MyNotificationTopic"
        }
    ],
    "Status": "ENABLED",
    "DetailType": "FULL"
}
```

儲存檔案。

3. 在終端機或命令列中，再次執行 create-notification-rule 命令，使用您剛編輯的檔案建立通知規則：

```
aws codestar-notifications create-notification-rule --cli-input-json
 file://rule.json
```

4. 如果成功，此命令會傳回通知規則的 ARN，如下所示：

```
{
    "Arn": "arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/
dc82df7a-EXAMPLE"
```

```
}
```

重命名 CodeDeploy 應用程式

您可以使用 AWS CLI 或 CodeDeploy API 來變更應用程式的名稱。

若要檢視應用程式名稱清單，請使用 AWS CLI 呼叫清單應用程式命令。

如需有關使用變更應用程 AWS CLI 式名稱的詳細資訊，請參閱更新應用程式。

如需使用 CodeDeploy API 變更應用程式名稱的相關資訊，請參閱 API UpdateApplication _。

在中刪除應用程式 CodeDeploy

您可以使用 CodeDeploy 主控台 AWS CLI、或 CodeDeploy API 動作來刪除應用程式。如需有關使用 CodeDeploy API 動作的資訊，請參閱DeleteApplication。

> **Warning**
>
> 刪除應用程式會從 CodeDeploy系統中移除應用程式的相關資訊，包括所有相關的部署群組資訊和部署詳細資訊。刪除針對 EC2/ 現場部署建立的應用程式不會從執行個體移除任何應用程式修訂，也不會從 Amazon S3 儲存貯體刪除修訂。刪除針對 EC2/ 現場部署建立的應用程式不會終止任何 Amazon EC2 執行個體或取消註冊任何現場部署執行個體。這個操作無法復原。

主題

- 刪除應用程式 (主控台)
- 刪除應用程式 (AWS CLI)

刪除應用程式 (主控台)

若要使用主 CodeDeploy 控台刪除應用程式：

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

> **Note**
>
> 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在瀏覽窗格中，展開 [部署]，然後選擇 [應用程式]。
3. 在應用程式清單中，選擇您要刪除的應用程式。

   會出現一個頁面，其中包含應用程式的詳細

4. 選擇右上角的刪除應用程序。
5. 出現提示時，輸入**delete**以確認您要刪除應用程式，然後選擇 [刪除]。

刪除應用程式 (AWS CLI)

若要使用 AWS CLI 來刪除應用程式，請呼叫 delete 應用程式命令，並指定應用程式名稱。若要檢視應用程式名稱清單，請呼叫清單應用程式命令。

使用中的部署群組 CodeDeploy

您可以為 CodeDeploy 應用程式指定一或多個部署群組。每個應用程式部署使用其中一個部署群組。部署群組包含部署期間使用的設定和組態。大部分的部署群組設定取決於應用程式所使用的運算平台。您可以為任何計算平台的部署群組設定某些設定，例如復原、觸發器和警示。

Amazon ECS 運算平台部署中的部署群組

在 Amazon ECS 部署中，部署群組會指定 Amazon ECS 服務、負載平衡器、選用的測試接聽程式和兩個目標群組。它還指定何時將流量重新路由到取代任務集，以及在成功部署後終止原始任務集和 Amazon ECS 應用程式的時機。

AWS Lambda 計算平台部署中的部署群組

在 AWS Lambda 部署中，部署群組會定義一組組 CodeDeploy態，以供 future 部署 AWS Lambda 函數使用。例如，部署群組指定如何將流量路由到新版本的 Lambda 函數。它也可以指定警示和轉返。AWS Lambda 部署群組中的單一部署可覆寫一或多個群組組態。

EC2/內部部署計算平台部署的部署群組

在 EC2 /內部部署中，部署群組是一組針對部署的個別執行個體。部署群組包含個別標記的執行個體、Amazon EC2 Auto Scaling 群組中的 Amazon EC2 執行個體，或兩者皆包含。

在就地部署中，部署群組中的執行個體會使用最新的應用程式修訂更新。

在藍/綠部署中，流量會從一個或多個負載平衡器中取消註冊原始執行個體，並註冊一組通常已安裝最新應用程式修訂版本的執行個體，將流量從一組執行個體重新路由傳送到另一個執行個體。

您可以將多個部署群組與中的應用程式相關聯 CodeDeploy。這可在不同時間內，將應用程式修訂部署到不同組執行個體。例如，您可能會使用一組部署群組，將應用程式修訂部署到一組套用 `Test` 標籤的執行個體，確認程式碼的品質。然後，您會將相同的應用程式修訂部署到包含套用 `Staging` 標籤之執行個體的部署群組，以做進一步的驗證。最後，當您準備好將最新的應用程式發行給客戶時，您會部署到包含套用 `Production` 標籤之執行個體的部署群組。

您也可以使用多個標籤群組，更進一步縮小包含在部署群組中執行個體的條件。如需相關資訊，請參閱Tagging Instances for Deployments。

當您使用 CodeDeploy 主控台建立應用程式時，您可以同時設定其第一個部署群組。當您使用建立應 AWS CLI 用程式時，請在單獨的步驟中建立其第一個部署群組。

若要檢視已與您的 AWS 帳戶相關聯的部署群組清單，請參閱檢視部署群組詳細資料 CodeDeploy。

如需 Amazon EC2 執行個體標籤的相關資訊，請參閱使用主控台使用標籤。如需內部部署執行個體的資訊，請參閱「Working with On-Premises Instances」。如需 Amazon EC2 Auto Scaling 的相關資訊，請參閱 CodeDeploy 與 Amazon EC2 Auto Scaling 集成。

主題

- the section called “建立部署群組”
- the section called “檢視部署群組詳情”
- the section called “變更部署群組設定”
- the section called “設定部署群組的進階選項”
- the section called “刪除部署群組”

建立部署群組 CodeDeploy

您可以使用 CodeDeploy 主控台 AWS CLI、 CodeDeploy API 或 AWS CloudFormation 範本來建立部署群組。如需使用 AWS CloudFormation 範本建立部署群組的相關資訊，請參閱AWS CloudFormation CodeDeploy供參考的範本。

當您使用 CodeDeploy 主控台建立應用程式時，您可以同時設定其第一個部署群組。當您使用建立應 AWS CLI 用程式時，請在單獨的步驟中建立其第一個部署群組。

做為建立部署群組的一部分，您必須指定服務角色。如需詳細資訊，請參閱 步驟 2：建立服務角色 CodeDeploy。

主題

- 建立就地部署的部署群組 (主控台)
- 為EC2/內部部署藍/綠部署建立部署群組 (主控台)
- 為 Amazon ECS 部署 (主控台) 建立部署群組
- 在 Elastic Load Balancing 中為 CodeDeploy Amazon EC2 部署設定負載平衡器
- 為 CodeDeploy Amazon ECS 部署設定負載平衡器、目標群組和接聽程式
- 建立部署群組 (CLI)

建立就地部署的部署群組 (主控台)

若要使用 CodeDeploy 主控台建立就地部署的部署群組：

> **Warning**
>
> 如果發生下列情況，請勿採用這些步驟：
>
> - 您尚未準備好要在應用程式首次 CodeDeploy部署中使用的執行個體。若要設定您的執行個體，請遵循使用的例證 CodeDeploy中的說明，然後遵循本主題中的步驟。
> - 您希望建立部署群組，使用自訂部署組態，但您尚未建立部署組態。請遵循Create a Deployment Configuration中的說明，再返回本主題中的步驟。
> - 您沒有至少信任信任中所述 CodeDeploy 的信任和權限的服務角色步驟 2：建立服務角色 CodeDeploy。若要建立及設定服務角色，請遵循步驟 2：建立服務角色 CodeDeploy中的說明，再返回本主題中的步驟。
> - 您想要在 Elastic Load Balancing 中為就地部署選取 Classic Load Balancer、應用程式負載平衡器或 Network Load Balancer，但尚未建立它。

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在瀏覽窗格中，展開 [部署]，然後選擇 [應用程式]。
3. 在 Applications (應用程式) 頁面上，選擇要建立部署群組的應用程式名稱。
4. 在您的應用程式頁面，從 Deployment groups (部署群組) 標籤中，選擇 Create deployment group (建立部署群組)。
5. 在 Deployment group name (部署群組名稱) 中，輸入描述部署群組的名稱。

   > **Note**
   >
   > 如果您想要使用在其他部署群組中使用的相同設定 (包括部署群組名稱、標籤、Amazon EC2 Auto Scaling 群組名稱或兩者；以及部署組態)，請在此頁面上指定這些設定。雖然這

個新的部署群組和現有的部署群組具有相同的名稱，但會 CodeDeploy 將它們視為個別的部署群組，因為它們都與不同的應用程式相關聯。

6. 在服務角色中，選擇授與目標執行個體 CodeDeploy存取權的服務角色。
7. 在 Deployment type (部署類型) 中，選擇 In-place (就地)。
8. 在環境組態中，執行下列動作：

   a. 如果您想要將應用程式部署到 Amazon EC2 Auto Scaling 群組，請選取 Amazon EC2 Auto Scaling 群組，然後選擇要將應用程式修訂部署到的 Amazon EC2 Auto Scaling 群組名稱。當新的 Amazon EC2 執行個體作為 Amazon EC2 自動擴展群組的一部分啟動時， CodeDeploy 可以自動將您的修訂部署到新執行個體。您最多可以將 10 個 Amazon EC2 Auto Scaling 群組新增至一個部署群組。如需詳細資訊，請參閱 CodeDeploy 與 Amazon EC2 Auto Scaling 集成。

   b. 如果您選取 Amazon EC2 Auto Scaling 群組，請選擇性地選取將終止勾點新增至自動擴展群組，以便在建立或更新部署群組時將終止勾點 CodeDeploy 安裝到 Auto Scaling 群組中。安裝此勾點後， CodeDeploy 將執行終止部署。如需詳細資訊，請參閱 在 Auto Scaling 擴充事件期間啟用終止部署。

   c. 如果要標記執行個體，請選取 Amazon EC2 執行個體或現場部署執行個體。在「機碼」和「值」欄位中，輸入用來標記例證的金鑰值配對值。您最多可以在單一標籤群組內標記 10 個金鑰值對。

      i. 您可以在「值」欄位中使用萬用字元來識別以特定模式標記的所有執行個體，例如類似的 Amazon EC2 執行個體、成本中心和群組名稱等。例如，如果您在「關鍵字」欄位中選擇「名稱」並**GRP-*a**在「值」欄位中輸入，則會 CodeDeploy 識別符合該陣列的所有例證**GRP-1a**，例如**GRP-2a**、和**GRP-XYZ-a**。

      ii. Value (值) 欄位區分大小寫。

      iii. 要從清單中移除一個鍵值組，請選擇移除圖示。

      CodeDeploy 尋找符合每個指定索引鍵值對或 Amazon EC2 Auto Scaling 群組名稱的執行個體時，會顯示相符執行個體的數量。要查看有關於執行個體的詳細資訊，請按一下數字。

      如果您想要強化部署到執行個體的條件，請選擇 Add tag group (新增標籤群組) 以建立標籤群組。您可以建立最多三個標籤群組，每個標籤群組最多包含 10 個金鑰值對。當您在部署群組中使用多個標籤群組，只有所有標籤群組識別出的執行個體會包含在部署群組中。這表示一個執行個體至少有一個標籤必須符合部署群組中的每個群組。

如需使用標籤群組來強化部署群組的相關資訊，請參閱Tagging Instances for Deployments。

9. 在使用 Systems Manager 的代理程式組態中，指定您希望如何在部署群組中的執行個體上安裝和更新 CodeDeploy 代理程式。如需 CodeDeploy 代理程式的詳細資訊，請參閱使用 CodeDeploy 代理程式。如需有關 Systems Manager 的詳細資訊，請參閱什麼是 Systems Manager？

   a. 永不：略過使用 Systems Manager 設定 CodeDeploy 安裝。執行個體必須安裝代理程式才能用於部署，因此只有在以其他方式安裝 CodeDeploy代理程式時，才選擇此選項。

   b. 只有一次：Systems Manager 會在部署群組中的每個執行個體上安裝 CodeDeploy 代理程式一次。

   c. 現在並排程更新：Systems Manager 會建立與狀態管理員的關聯，依照您設定的排程安裝 CodeDeploy 代理程式。有關狀態管理員和關聯的詳細資訊，請參閱關於狀態管理員。

10. 在 Deployment configuration (部署組態) 中，選擇部署組態以控制執行個體的部署速度，例如一次一個或一次全部。如需部署組態的詳細資訊，請參閱使用中的部署組態 CodeDeploy。

11. (選擇性) 在負載平衡器中，選取啟用負載平衡，然後從清單中選取傳統負載平衡器、Application Load Balancer 目標群組和 Network Load Balancer 目標群組，以便在 CodeDeploy 部署期間管理執行個體的流量。您最多可以選取 10 個傳統負載平衡器和 10 個目標群組，總共 20 個項目。確保要部署到的 Amazon EC2 執行個體已向所選負載平衡器 (傳統負載平衡器) 或目標群組 (應用程式負載平衡器和網路負載平衡器) 註冊。

    在部署期間，原始執行個體會從選取的負載平衡器和目標群組取消註冊，以防止在部署期間將流量路由傳送到這些執行個體。部署完成後，每個執行個體都會重新註冊所有選取的傳統負載平衡器和目標群組。

    如需 CodeDeploy 部署負載平衡器的詳細資訊，請參閱Integrating CodeDeploy with Elastic Load Balancing。

    > **Warning**
    >
    > 如果您在此部署群組中同時設定 Auto Scaling 群組和 Elastic Load Balancing 負載平衡器，並且想要將負載平衡器連接至 Auto Scaling 群組，建議您在從此部署群組建立 CodeDeploy 部署之前完成此附件。在建立部署後嘗試完成附件，可能會導致所有執行個體意外地從負載平衡器中取消註冊。

12. (選擇性) 展開進階並設定您要包含在部署中的任何選項，例如 Amazon SNS 通知觸發器、Amazon CloudWatch 警示、Auto Scaling 展選項或自動復原。

    如需詳細資訊，請參閱 設定部署群組的進階選項。

13. 選擇 Create deployment group (建立部署群組)。

為EC2/內部部署藍/綠部署建立部署群組 (主控台)

若要使用 CodeDeploy 主控台為藍/綠部署建立部署群組：

> **Warning**
>
> 如果發生下列情況，請勿採用這些步驟：
>
> - 在藍/綠部署程序期間，您沒有安裝 CodeDeploy 代理程式的執行個體要取代。若要設定您的執行個體，請遵循使用的例證 CodeDeploy中的說明，然後遵循本主題中的步驟。
> - 您希望建立使用自訂部署組態的應用程式，但您尚未建立部署組態。請遵循Create a Deployment Configuration中的說明，再返回本主題中的步驟。
> - 您沒有至少信任信任中所述 CodeDeploy 的信任和權限的服務角色步驟 2：建立服務角色 CodeDeploy。若要建立及設定服務角色，請遵循步驟 2：建立服務角色 CodeDeploy中的說明，再返回本主題中的步驟。
> - 您尚未在 Elastic Load Balancing 中建立 Classic Load Balancer 或應用程式負載平衡器來註冊替代環境中的執行個體。如需詳細資訊，請參閱 在 Elastic Load Balancing 中為 CodeDeploy Amazon EC2 部署設定負載平衡器。

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在瀏覽窗格中，展開 [部署]，然後選擇 [應用程式]。
3. 在 Applications (應用程式) 頁面上，選擇要建立部署群組的應用程式名稱。
4. 在您的應用程式頁面，從 Deployment groups (部署群組) 標籤中，選擇 Create deployment group (建立部署群組)。
5. 在 Deployment group name (部署群組名稱) 中，輸入描述部署群組的名稱。

> **Note**
>
> 如果您想要使用在其他部署群組中使用的相同設定 (包括部署群組名稱、標籤、Amazon EC2 Auto Scaling 群組名稱和部署組態)，請在此頁面上選擇這些設定。雖然這個新的部署群組和現有的部署群組具有相同的名稱，但會 CodeDeploy 將它們視為個別的部署群組，因為它們與不同的應用程式相關聯。

6. 在服務角色中，選擇授與目標執行個體 CodeDeploy存取權的服務角色。
7. 在 Deployment type (部署類型) 中，選擇 Blue/green (藍/綠)。
8. 在環境組態中，執行下列動作：

   - 選取要用來為您的取代環境提供執行個體的方法。您有下列選項：
     - 自動複製 Amazon EC2 自 Auto Scaling 群組： CodeDeploy透過複製您指定的群組來建立 Amazon EC2 Auto Scaling 群組。
     - 手動佈建執行個體：直到建立部署，您才能為您的替換環境指定執行個體。開始部署之前，您必須建立執行個體。在這個選項中，您要改為指定欲取代的執行個體。
   - 如果您選取了「自動複製 Amazon EC2 Auto Scaling 群組」，請選擇性地選取「將終止勾點新增至 Auto Scaling 群組」，以便在建立或更新部署群組時將終止勾點 CodeDeploy 安裝到 Auto Scaling 群組。安裝此勾點後， CodeDeploy 將執行終止部署。如需詳細資訊，請參閱 在 Auto Scaling 擴充事件期間啟用終止部署。
9. 在使用 Systems Manager 的代理程式組態中，指定您希望如何在部署群組中的執行個體上安裝和更新 CodeDeploy 代理程式。如需 CodeDeploy 代理程式的詳細資訊，請參閱使用 CodeDeploy 代理程式。如需有關 Systems Manager 的詳細資訊，請參閱什麼是 Systems Manager？

   a. 永不：略過使用 Systems Manager 設定 CodeDeploy 安裝。執行個體必須安裝代理程式才能用於部署，因此只有在以其他方式安裝 CodeDeploy代理程式時，才選擇此選項。

   b. 只有一次：Systems Manager 會在部署群組中的每個執行個體上安裝 CodeDeploy 代理程式一次。

   c. 現在並排程更新：Systems Manager 會建立與狀態管理員的關聯，依照您設定的排程安裝 CodeDeploy 代理程式。有關狀態管理員和關聯的詳細資訊，請參閱關於狀態管理員。
10. 根據您在步驟 8 中所做的選擇，執行下列任一作業：

    - 如果您選擇了自動複製 Amazon EC2 Auto Scaling 群組：在 Amazon EC2 Auto Scaling 群組中，選擇或輸入您要用作為替代環境中執行個體建立之 Amazon EC2 Auto Scaling 群組範本

的 Amazon EC2 自動擴展群組的名稱。您選取的 Amazon EC2 Auto Scaling 群組中目前運作良好的執行個體數目是在替代環境中建立的。

- 如果您選擇手動佈建執行個體：選取 Amazon EC2 Auto Scaling 群組、Amazon EC2 Auto Scaling 安全性或兩者，以指定要新增至此部署群組的執行個體。輸入 Amazon EC2 Auto Scaling 標籤值或 Amazon EC2 Auto Scaling 群組名稱，以識別原始環境中的執行個體 (也就是您要取代或執行目前應用程式修訂版的執行個體)。

11. 在負載平衡器中，選取啟用負載平衡，然後從清單中選取要向其註冊替代 Amazon EC2 執行個體的傳統負載平衡器、應用程式負載平衡器目標群組和 Network Load Balancer 目標群組。每個取代執行個體都會向所有選取的傳統負載平衡器和目標群組登錄。您最多可以選取 10 個傳統負載平衡器和 10 個目標群組，總共 20 個項目。

    系統會根據您選擇的流量重新路由傳送和部署組態設定，將流量從原始執行個體重新路由傳送至替代執行個體。

    如需 CodeDeploy 部署負載平衡器的詳細資訊，請參閱Integrating CodeDeploy with Elastic Load Balancing。

    > ⚠ Warning
    >
    > 如果您在此部署群組中同時設定 Auto Scaling 群組和 Elastic Load Balancing 負載平衡器，並且想要將負載平衡器連接至 Auto Scaling 群組，建議您先完成此附件，然後再從此 CodeDeploy 部署群組建立部署。在建立部署後嘗試完成附件，可能會導致所有執行個體意外地從負載平衡器中取消註冊。

12. 在 Deployment settings (部署設定) 中，檢閱重新路由流量至替換環境的預設選項、要用於部署的部署組態，以及部署後處理原始環境中執行個體的方式。

    若您想要變更設定，請繼續下一個步驟。否則，請跳至步驟 14。

13. 若要變更藍/綠部署的部署設定，請選擇下列任何設定。

設定	選項
Traffic rerouting (重新路由流量)	• 立即重新路由流量：一旦在取代環境中佈建執行個體並在其上安裝了最新的應用程式修訂版本，就會自動向指定的負載平衡器和目標群組登錄這些執行個體，進而導致流量重

設定	選項
	新路由傳送給這些執行個體。然後撤銷註冊原始環境中的執行個體。 • 我將選擇是否重新路由傳送流量：除非您手動重新路由傳送流量，否則取代環境中的執行個體不會向指定的負載平衡器和目標群組登錄。如果過了您指定的等待時間卻沒有重新路由流量，則部署狀態會變更為「已停止」。
Deployment configuration (部署組態)	選擇替代環境中的執行個體向負載平衡器和目標群組註冊的速率，例如一次登錄一個執行個體或一次全部登錄。 **Note** 流量成功路由到替換環境之後，無論選取哪一種部署設定，都會立即將原始環境中的執行個體全部撤銷註冊。 如需詳細資訊，請參閱 使用中的部署組態 CodeDeploy。
Original instances (原始執行個體)	• 終止部署群組中的原始執行個體：流量重新路由傳送至取代環境後，從負載平衡器和目標群組取消註冊的執行個體會在您指定的等待期間後終止。 • 讓部署群組中的原始執行個體保持執行中：將流量重新路由傳送至取代環境後，從負載平衡器和目標群組取消註冊的執行個體會保持在執行狀態。

14. (選擇性) 在進階中，設定要包含在部署中的選項，例如 Amazon SNS 通知觸發器、Amazon CloudWatch 警示、Auto Scaling 選項或自動復原。

    如需在部署群組中指定進階選項的相關資訊，請參閱設定部署群組的進階選項。

15. 選擇 Create deployment group (建立部署群組)。

為 Amazon ECS 部署 (主控台) 建立部署群組

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在瀏覽窗格中，展開 [部署]，然後選擇 [應用程式]。
3. 在 Applications table (應用程式資料表) 中，選擇與您想編輯之部署群組相關聯的應用程式名稱。
4. 在應用程式頁面的 Deployment groups (部署群組) 上，選擇您想編輯的部署群組名稱。
5. 在您的應用程式頁面，從 Deployment groups (部署群組) 標籤中，選擇 Create deployment group (建立部署群組)。如需針對 Amazon ECS 部署建立部署群組所需項目的詳細資訊，請參閱在您開始 Amazon ECS 部署之前。
6. 在 Deployment group name (部署群組名稱) 中，輸入描述部署群組的名稱。

   > **Note**
   >
   > 如果您想使用用於其他部署群組的相同設定 (包括部署群組名稱和部署組態)，請在本頁面上選擇這些設定。雖然這個新群組和現有群組可能具有相同的名稱，但是會 CodeDeploy 將它們視為個別的部署群組，因為每個群組都與個別的應用程式相關聯。

7. 在服務角色中，選擇授與 Amazon ECS CodeDeploy 存取權的服務角色。如需詳細資訊，請參閱步驟 2：建立服務角色 CodeDeploy。
8. 從負載平衡器名稱中，選擇為 Amazon ECS 服務提供流量的負載平衡器名稱。
9. 從生產接聽程式連接埠中，選擇為向 Amazon ECS 服務提供生產流量的接聽程式的連接埠和通訊協定。
10. (選擇性) 從測試接聽程式連接埠中，選擇測試接聽程式的連接埠和通訊協定，以在部署期間為 Amazon ECS 服務中設定的替換任務提供流量。您可以在`AfterAllowTestTraffic`掛接期間執行的 AppSpec 檔案中指定一或多個 Lambda 函數。這些函數可以運行驗證測試。如果驗證測試失敗，則會觸發部署復原。如果驗證測試成功，則會觸發部署生命週期中的下一個勾點`BeforeAllowTraffic`。如果未指定測試接聽程式連接埠，則`AfterAllowTestTraffic`掛接期間不會發生任何事情。如需詳細資訊，請參閱 AppSpec Amazon ECS 部署的「掛鉤」部分。

11. 從目標群組 1 名稱和目標群組 2 名稱中，選擇部署期間用於路由傳送流量的目標群組。CodeDeploy 將一個目標群組繫結至 Amazon ECS 服務的原始任務集，另一個目標群組繫結至其替換任務集。如需詳細資訊，請參閱應用程式負載平衡器的目標群組。
12. 選擇「立即重新路由流量」或「指定何時重新路由流量」，以確定何時將流量重新路由至更新的 Amazon ECS 服務。

    如果您選擇「立即重新路由傳送流量」，則部署會在佈建取代工作集後自動重新路由傳送流量。

    如果您選擇指定重新路由傳送流量的時間，請選擇成功佈建取代工作集後要等待的天數、小時數和分鐘數。在此等待時間內，會執行 AppSpec 檔案中指定的 Lambda 函數中的驗證測試。如果等待時間在重新路由傳送流量之前過期，則部署狀態會變更為`Stopped`。
13. 對於原始修訂終止，請選擇在成功部署後等待的天數、小時數和分鐘數，然後在 Amazon ECS 服務中設定的原始任務終止。
14. (選擇性) 在進階中，設定要包含在部署中的任何選項，例如 Amazon SNS 通知觸發器、Amazon CloudWatch 警示或自動復原。

    如需詳細資訊，請參閱 設定部署群組的進階選項。

在 Elastic Load Balancing 中為 CodeDeploy Amazon EC2 部署設定負載平衡器

在執行任何藍/綠部署或要在部署群組中指定選用性負載平衡器的就地部署之前，您必須在 Elastic Load Balancing 中至少建立一個 Classic Load Balancer、Application Load Balancer 載平衡器或 Network Load Balancer。針對藍色/綠色部署，您會使用該負載平衡器註冊執行個體，組成您的取代環境。您原始環境中的執行個體可選擇性地向此相同負載平衡器註冊。對於就地部署，負載平衡器可用來取消註冊正在處理的執行個體 CodeDeploy，並在工作完成時重新註冊這些執行個體。

CodeDeploy 支援藍/綠和就地部署到多個負載平衡器後方的 Amazon EC2 執行個體。例如，假設您有 200 個 Amazon EC2 執行個體，其中 100 個執行個體向 2 個傳統負載平衡器註冊，而另外 100 個執行個體則向 2 個應用程式負載平衡器中的 4 個目標群組註冊。在此案例中，可讓 CodeDeploy 您對所有 200 個執行個體進行藍/綠和就地部署，即使這些執行個體分散在 2 個傳統負載平衡器、2 個應用程式負載平衡器和 4 個目標群組。

CodeDeploy 最多支援 10 個傳統負載平衡器和 10 個目標群組，總共 20 個項目。

若要設定一或多個 Classic Load Balancer，請遵循傳統負載平衡器使用者指南中的教學課程：建立傳統負載平衡器中的指示。注意下列事項：

- 在步驟 2：定義負載平衡器中，在於內部建立 LB 內，選擇您在建立執行個體時選取的相同 VPC。
- 在步驟 5：向您的負載平衡器註冊 EC2 執行個體中，選取目前位於您部署群組 (就地部署) 中的執行個體，或是您已指定位於原始環境 (藍色/綠色部署) 中的執行個體。
- 在步驟 7：建立並確認您的負載平衡器中，記下您負載平衡器的 DNS 地址。

  例如，若您將您的負載平衡器命名為 `my-load-balancer`，您的 DNS 地址會以類似下列格式出現：`my-load-balancer-1234567890.us-east-2.elb.amazonaws.com`。

若要設定一或多個應用程式負載平衡器，請遵循下列其中一個主題中的指示：

- 建立應用程式負載平衡器
- 教學課程：使用建立應用程式負載平衡器 AWS CLI

若要設定一或多個網路負載平衡器，請遵循下列其中一個主題中的指示：

- 建立 Network Load Balancer
- 教學課程：使用建立 Network Load Balancer AWS CLI

為 CodeDeploy Amazon ECS 部署設定負載平衡器、目標群組和接聽程式

在使用 Amazon ECS 運算平台執行部署之前，您必須建立 Application Load Balancer 或 Network Load Balancer、兩個目標群組以及一或兩個接聽程式。本主題說明如何建立應用程式負載平衡器。如需詳細資訊，請參閱 在您開始 Amazon ECS 部署之前。

其中一個目標群組會將流量導向 Amazon ECS 應用程式的原始任務集。另一個目標群組會將流量導向其替換任務集。在部署期間， CodeDeploy 會建立取代工作集，並將原始工作集的流量重新路由傳送至新工作集。 CodeDeploy 決定每個作業集使用哪個目標群組。

接聽程式供負載平衡器用來將流量導向到您的目標群組。一個生產接聽程式為必要項目。您可以指定選用的測試接聽程式，負責在您執行驗證測試時引導流量到您的替換任務。

負載平衡器必須使用 VPC 搭配不同可用區域中的兩個子網路。以下步驟說明如何確認預設 VPC、建立 Amazon EC2 Application Load Balancer，然後為負載平衡器建立兩個目標群組。如需詳細資訊，請參閱網路負載平衡器的目標群組。

驗證您的預設 VPC、公用子網路和安全性群組

本主題說明如何建立 Amazon EC2 Application Load Balancer、兩個目標群組以及兩個可在 Amazon ECS 卸載期間使用的連接埠。其中一個連接埠是選用的，且只有在部署期間將流量導向測試連接埠進行驗證測試時才需要。

1. 登入 AWS Management Console 並開啟 Amazon VPC 主控台，網址為 https://console.aws.amazon.com/vpc/。
2. 驗證要使用的預設 VPC。在導覽窗格中，選擇 Your VPCs (您的 VPC)。請注意，哪個 VPC 在 Default VPC (預設 VPC) 欄中顯示 Yes (是)。這是您的預設 VPC。其中包含您使用的預設子網路。
3. 選擇 Subnets (子網路)。記下在 Default subnet (預設子網路) 欄中顯示 Yes (是) 的兩個子網路的子網路 ID。當您建立負載平衡器時會使用這些 ID。
4. 選擇每個子網路，然後選擇 Description (描述) 標籤。驗證您想要使用的子網路是否位於不同的可用區域中。
5. 選擇子網路，然後選擇 Route Table (路由表) 標籤。若要驗證您想要使用的每個子網路是否為公有子網路，請確認路由表中有一列具有網際網路閘道的連結。
6. 登入 AWS Management Console 並開啟 Amazon EC2 主控台，網址為 https://console.aws.amazon.com/ec2/。
7. 從導覽窗格中，選擇 Security Groups (安全群組)。
8. 確認您想要使用的安全群組可用，並記下其群組 ID (例如，sg-abcd1234)。您在建立負載平衡器時會使用此項目。

建立 Amazon EC2 Application Load Balancer、兩個目標群組和接聽程式 (主控台)

若要使用 Amazon EC2 主控台建立 Amazon EC2 Application Load Balancer：

1. 登入 AWS Management Console 並開啟 Amazon EC2 主控台，網址為 https://console.aws.amazon.com/ec2/。
2. 在導覽窗格中，選擇 Load Balancers (負載平衡器)。
3. 選擇 Create Load Balancer (建立負載平衡器)。
4. 選擇 Application Load Balancer (應用程式負載平衡器)，然後選擇 Create (建立)。
5. 在 Name (名稱) 中，輸入負載平衡器的名稱。
6. 在 Scheme (結構描述) 中，選擇 internet-facing (面向網際網路)。

7. 在 IP address type (IP 地址) 中，選擇 ipv4。
8. (選用) 為您的負載平衡器設定第二個接聽程式連接埠。您可以使用轉發至此連接埠的測試流量來執行部署驗證測試。

   a. 在 Load Balancer Protocol (負載平衡器通訊協定) 下，選擇 Add listener (新增接聽程式)。

   b. 在第二個接聽程式的 Load Balancer Protocol (負載平衡器通訊協定) 下，選擇 HTTP。

   c. 在 Load Balancer Port (負載平衡器連接埠) 下，輸入 **8080**。
9. 在 Availability Zones (可用區域) 下的 VPC 中，選擇預設 VPC，然後選取您要使用的兩個預設子網路。
10. 選擇 Next: Configure Security Settings (下一步：設定安全設定)。
11. 選擇 Next: Configure Security Groups (下一步：設定安全群組)。
12. 選擇 Select an existing security group (選取現有的安全群組)，選擇預設安全群組，然後記下其 ID。
13. 選擇 Next: Configure Routing (下一步：設定路由)。
14. 在 Target group (目標群組) 中，選擇 New target group (新增目標群組)，然後設定您的第一個目標群組：

    a. 在 Name (名稱) 中，輸入目標群組名稱 (例如，**target-group-1**)。

    b. 在 Target type (目標類型) 中，選擇 IP。

    c. 在 Protocol (通訊協定) 中，選擇 HTTP。在 Port (連接埠) 中，輸入 **80**。

    d. 選擇 Next: Register Targets (下一步：註冊目標)。
15. 選擇 Next: Review (下一步：檢視)，然後選擇 Create (建立)。

為您的負載平衡器建立第二個目標群組

1. 佈建負載平衡器後，開啟 Amazon EC2 主控台。在導覽窗格中，選擇 Target Groups (目標群組)。
2. 選擇 Create target group (建立目標群組)。
3. 在 Name (名稱) 中，輸入目標群組名稱 (例如，**target-group-2**)。
4. 在 Target type (目標類型) 中，選擇 IP。
5. 在 Protocol (通訊協定) 中，選擇 HTTP。在 Port (連接埠) 中，輸入 **80**。
6. 在 VPC 中，選擇預設 VPC。
7. 選擇建立。

> **Note**
>
> 您必須為負載平衡器建立兩個目標群組，才能執行 Amazon ECS 部署。建立 Amazon ECS 服務時，您可以使用其中一個目標群組的 ARN。如需詳細資訊，請參閱 Amazon ECS 使用者指南中的步驟 4：建立 Amazon ECS 服務。

建立 Amazon EC2 Application Load Balancer、兩個目標群組和接聽程式 (CLI)

若要使用建立應用程式負載平衡器 AWS CLI：

1. 使用命create-load-balancer令建立應用程式負載平衡器。指定兩個不在相同可用區域中的子網路和一個安全群組。

   ```
   aws elbv2 create-load-balancer --name bluegreen-alb \
   --subnets subnet-abcd1234 subnet-abcd5678 --security-groups sg-abcd1234 --
   region us-east-1
   ```

   輸出包含負載平衡器的 Amazon Resource Name (ARN)，格式如下：

   ```
   arn:aws:elasticloadbalancing:region:aws_account_id:loadbalancer/app/bluegreen-alb/
   e5ba62739c16e642
   ```

2. 使用指create-target-group令建立第一個目標群組。 CodeDeploy 將此目標群組的流量路由至服務中設定的原始或取代工作。

   ```
   aws elbv2 create-target-group --name bluegreentarget1 --protocol HTTP --port 80 \
   --target-type ip --vpc-id vpc-abcd1234 --region us-east-1
   ```

   輸出包含第一個目標群組的 ARN，格式如下：

   ```
   arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
   bluegreentarget1/209a844cd01825a4
   ```

3. 使用指create-target-group令建立第二個目標群組。 CodeDeploy 將目標群組的流量路由至第一個目標群組未提供服務的任務集。例如，如果您的第一個目標群組將流量路由到原始任務集，此目標群組會將流量路由到替換任務集。

```
aws elbv2 create-target-group --name bluegreentarget2 --protocol HTTP --port 80 \
--target-type ip --vpc-id vpc-abcd1234 --region us-east-1
```

輸出包含第二個目標群組的 ARN，格式如下：

```
arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
bluegreentarget2/209a844cd01825a4
```

4. 使用 create-listener 命令，以預設規則建立接聽程式，將生產流量轉送至 8080 埠。

```
aws elbv2 create-listener --load-balancer-arn
 arn:aws:elasticloadbalancing:region:aws_account_id:loadbalancer/app/bluegreen-alb/
e5ba62739c16e642 \
--protocol HTTP --port 80 \
--default-actions
 Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup
bluegreentarget1/209a844cd01825a4 --region us-east-1
```

輸出包含接聽程式的 ARN，格式如下：

```
arn:aws:elasticloadbalancing:region:aws_account_id:listener/app/bluegreen-alb/
e5ba62739c16e642/665750bec1b03bd4
```

5. (選用) 使用 create-listener 命令，以預設規則建立第二個接聽程式，將測試流量轉送至 8080 埠。您可以使用轉發至此連接埠的測試流量來執行部署驗證測試。

```
aws elbv2 create-listener --load-balancer-arn
 arn:aws:elasticloadbalancing:region:aws_account_id:loadbalancer/app/bluegreen-alb/
e5ba62739c16e642 \
--protocol HTTP --port 8080 \
--default-actions
 Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup
bluegreentarget2/209a844cd01825a4 --region us-east-1
```

輸出包含接聽程式的 ARN，格式如下：

```
arn:aws:elasticloadbalancing:region:aws_account_id:listener/app/bluegreen-alb/
e5ba62739c16e642/665750bec1b03bd4
```

建立部署群組 (CLI)

若要使用 AWS CLI 建立部署群組，請呼叫命create-deployment-group令，並指定：

- 應用程式名稱。若要檢視應用程式名稱清單，請呼叫清單應用程式命令。
- 部署群組的名稱。系統會針對指定的應用程式建立具有此名稱的部署群組。此外，部署群組僅能與一個應用程式建立關聯。
- 標籤、標籤群組或 Amazon EC2 Auto Scaling 群組名稱的相關資訊，這些群組可識別要包含在部署群組中的執行個體。
- 服務角色的 Amazon 資源名稱 (ARN) 識別碼，可 CodeDeploy 在與其他 AWS 服務互動時代表您的 AWS 帳戶執行動作。如需取得服務角色的 ARN，請參閱取得服務角色 ARN (CLI) 。如需有關服務角色的詳細資訊，請參閱 IAM 使用者指南中的角色術語和概念。
- 要與部署群組建立關聯的部署類型 (就地或藍/綠) 的相關資訊。
- (選用) 現有部署組態的名稱。若要檢視部署組態清單，請參閱View Deployment Configuration Details。如果未指定，則 CodeDeploy 使用預設部署規劃。
- (選用) 建立觸發器的命令，將有關部署和執行個體事件的通知推送給訂閱 Amazon 簡單通知服務主題的使用者。如需詳細資訊，請參閱 Monitoring Deployments with Amazon SNS Event Notifications。
- (選擇性) 在 CloudWatch 警示中指定的測量結果低於或超過定義的臨界值時，將現有警示新增至已啟動的部署群組的命令。
- (選擇性) 當部署失敗或啟動 CloudWatch 警示時，用於復原至上次已知的正確修訂版本的指令。
- (選擇性) 用於在 Auto Scaling 事件期間產生生命週期事件掛接的部署指令。如需詳細資訊，請參閱亞馬遜 EC2 Auto Scaling 如何與 CodeDeploy。
- 針對就地部署：
  - (選擇性) Elastic Load Balancing 中的傳統負載平衡器、應用程式負載平衡器或網路負載平衡器的名稱，用於在部署程序期間管理執行個體的流量。
- 針對藍色/綠色部署：
  - 藍色/綠色部署程序組態：
    - 取代環境中新執行個體的佈建方式。
    - 是否要立即將流量重新路由至取代環境，或是在指定期間內等待手動重新路由流量。
    - 是否要終止原始環境中的執行個體。
  - Elastic Load Balancing 中用於在替代環境中註冊的執行個體的傳統負載平衡器、應用程式負載平衡器或網路負載平衡器的名稱。

> **Warning**
>
> 如果您要在部署群組中同時設定 Auto Scaling 群組和 Elastic Load Balancing 器，並且想要將負載平衡器連結至 Auto Scaling 群組，建議您先完成此附件，然後再從此 CodeDeploy 部署群組建立部署。在建立部署後嘗試完成附件，可能會導致所有執行個體意外地從負載平衡器取消註冊。

檢視部署群組詳細資料 CodeDeploy

您可以使用主 CodeDeploy 控台 AWS CLI、或 CodeDeploy API 來檢視與應用程式相關聯之所有部署群組的詳細資料。

主題

- 檢視部署群組詳細資料 (主控台)
- 檢視部署群組詳細資料 (CLI)

檢視部署群組詳細資料 (主控台)

若要使用主 CodeDeploy 控台檢視部署群組詳細資料：

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在瀏覽窗格中，展開 [部署]，然後選擇 [應用程式]。
3. 在 Applications (應用程式) 頁面上，選擇與部署群組相關聯的應用程式名稱。

   > **Note**
   >
   > 如果未顯示任何項目，請確定已選取正確的區域。在導覽列的區域選取器中，選擇「區域」和「端點」中列出的其中一個區域AWS 一般參考。 CodeDeploy 僅在這些地區支援。

4. 若要檢視個別部署群組的詳細資訊，請在 Deployment groups (部署群組) 標籤中，選擇部署群組的名稱。

檢視部署群組詳細資料 (CLI)

若要使用 AWS CLI 檢視部署群組詳細資料，請呼叫`get-deployment-group`命令或命`list-deployment-groups`令。

若要檢視單一部署群組的詳細資料，請呼叫命get-deployment-group令，並指定：

- 與部署群組建立關聯的應用程式名稱。若要取得應用程式名稱，請呼叫清單應用程式命令。
- 部署群組名稱。若要取得部署群組名稱，請呼叫命list-deployment-groups令。

若要檢視部署群組名稱清單，請呼叫list-deployment-groups命令，指定與部署群組相關聯的應用程式名稱。若要取得應用程式名稱，請呼叫清單應用程式命令。

變更部署群組設定 CodeDeploy

您可以使用 CodeDeploy 控制台 AWS CLI、或 CodeDeploy API 來變更部署群組的設定。

> **Warning**
>
> 如果您希望部署群組使用 not-yet-created自訂部署群組，請勿使用這些步驟。請改為遵循Create a Deployment Configuration中的說明，再返回本主題。如果您希望部署群組使用不同的 not-yet-created 服務角色，請勿使用這些步驟。服務角色至少必須信任 CodeDeploy 中所述的權限步驟 2：建立服務角色 CodeDeploy。若要使用正確的許可建立及設定服務角色，請遵循步驟 2：建立服務角色 CodeDeploy中的說明，再返回本主題。

主題

變更部署群組設定 (主控台)

如果要使用 CodeDeploy 主控台變更部署群組設定：

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在瀏覽窗格中，展開 [部署]，然後選擇 [應用程式]。
3. 在應用程式清單中，選擇與您欲變更部署群組建立關聯的應用程式名稱。

   > **Note**
   > 如果未顯示任何項目，請確定已選取正確的區域。在導覽列的區域選取器中，選擇「區域」和「端點」中列出的其中一個區域AWS 一般參考。 CodeDeploy 僅在這些地區支援。

4. 選擇 Deployment groups (部署群組) 標籤，然後選擇您欲變更的部署群組名稱。
5. 在 Deployment group (部署群組) 頁面上，選擇 Edit (編輯)。
6. 請視需要編輯部署群組選項。

   如需部署群組元件的相關資訊，請參閱建立部署群組 CodeDeploy。

7. 選擇儲存變更。

變更部署群組設定 (CLI)

若要使用變 AWS CLI 更部署群組設定，請呼叫命update-deployment-group令，並指定：

- 對於 EC2 /內部部署和 AWS Lambda 部署：
  - 應用程式名稱。若要檢視應用程式名稱清單，請呼叫清單應用程式命令。
  - 目前的部署群組名稱。若要檢視部署群組名稱清單，請呼叫命list-deployment-groups令。
  - (選擇性) 不同的部署群組名稱。
  - (選擇性) 不同的 Amazon 資源名稱 (ARN) 對應至服務角色，可在與其他服 AWS 務互動時代表您的 AWS 帳戶執行動作。 CodeDeploy 如需取得服務角色的 ARN，請參閱取得服務角色 ARN (CLI)。如需有關服務角色的詳細資訊，請參閱 IAM 使用者指南中的角色術語和概念。
  - (選擇性) 部署組態的名稱。若要檢視部署組態清單，請參閱View Deployment Configuration Details。(如果未指定，則 CodeDeploy 會使用預設部署規劃。)

  - (選擇性) 在 CloudWatch 警示中指定的測量結果低於或超過定義的臨界值時，將一或多個現有警示新增至已啟動的部署群組的命令。
  - (選擇性) 當部署失敗或啟動 CloudWatch 警示時，用於復原至上次已知的正確修訂版本的指令。
  - (選擇性) 用於在 Auto Scaling 事件期間產生生命週期事件掛接的部署指令。如需詳細資訊，請參閱 亞馬遜 EC2 Auto Scaling 如何與 CodeDeploy。
  - (選擇性) 用於建立或更新發佈至 Amazon Simple Notification Service 主題的觸發器的命令，讓該主題的訂閱者會收到有關此部署群組中部署和執行個體事件的通知。如需相關資訊，請參閱Monitoring Deployments with Amazon SNS Event Notifications。
- 僅適用於 EC2 /內部部署：
  - (選擇性) 取代標籤或標籤群組，用來唯一識別要包含在部署群組中的執行個體。
  - (選擇性) 要新增至部署群組的取代 Amazon EC2 Auto Scaling 群組名稱。
- 僅適用於 Amazon ECS 部署：
  - 要部署的 Amazon ECS 服務。
  - 負載平衡器資訊，包括 Application Load Balancer 或 Network Load Balancer、Amazon ECS 部署所需的目標群組，以及生產和選用的測試接聽程式資訊。

設定部署群組的進階選項

當您建立或更新部署群組時，您可以設定數個選項，針對該部署群組的部署提供更多控制及監督。

在您使用下列主題中的部署群組時，使用此頁面上的資訊協助您設定進階選項：

- 建立應用程式 CodeDeploy
- 建立部署群組 CodeDeploy
- 變更部署群組設定 CodeDeploy

Amazon SNS 通知觸發器：您可以將觸發器新增至 CodeDeploy 部署群組，以接收與該部署群組中部署相關事件的通知。這些通知會傳送給訂閱您已參與觸發器動作一部分之 Amazon SNS 主題的收件者。

您必須已設定此觸發器將指向的 Amazon SNS 主題，並且 CodeDeploy必須具有從此部署群組發佈至主題的權限。若您尚未完成這些設定步驟，您可以稍後再將觸發新增到部署群組。

若您想要立即建立觸發，以接收此應用程式部署群組中部署事件的通知，請選擇建立觸發。

如果您的部署是 Amazon EC2 執行個體，您可以建立執行個體的通知，並接收有關執行個體的通知。

如需詳細資訊，請參閱 Monitoring Deployments with Amazon SNS Event Notifications。

Amazon CloudWatch 警示：您可以建立 CloudWatch 警示來監視指定時段內的單一指標，並根據指定臨界值在多個時段內相對於指定閾值的指標值執行一或多個動作。對於 Amazon EC2 部署，您可以為您在 CodeDeploy 操作中使用的執行個體或 Amazon EC2 Auto Scaling 群組建立警示。對於 AWS Lambda 和 Amazon ECS 部署，您可以針對 Lambda 函數中的錯誤建立警示。

您可以將部署設定為在 Amazon CloudWatch 警示偵測到指標低於或超過定義的閾值時停止。

您必須 CloudWatch 先在中建立警示，才能將其新增至部署群組。

1. 若要將警示監控新增至部署群組，請在 Alarms (警示) 中選擇 Add alarm (新增警示)。
2. 輸入您已設定為監視此部署的 CloudWatch 警示名稱。

   您必須輸入 CloudWatch 鬧鐘與中建立的警示完全相同 CloudWatch。若要檢視警示清單，請在的開啟 CloudWatch 主控台https://console.aws.amazon.com/cloudwatch/，然後選擇 [鬧鐘]。

其他選項：

- 若您想要繼續部署，而無需考慮您新增的警示，請選擇 Ignore alarm configuration (忽略警示組態)。

  這個選擇在您希望暫時停用部署群組的警示監控，又不需要稍後再次新增相同的警示時非常有用。
- (選擇性) 如果您希望在無法從 Amazon 擷取警示狀態的 CodeDeploy 事件中繼續進行部署 CloudWatch，請選擇即使警示狀態無法使用仍繼續部署。

> **Note**
>
> 此選項對ignorePollAlarmFailure應於 CodeDeploy API 中的AlarmConfiguration物件中。

如需詳細資訊，請參閱 使用 CloudWatch 警示監控部署 CodeDeploy。

自動轉返：您可以設定部署群組，或設定部署在部署失敗或到達您指定的監控閾值時自動轉返。在此案例下，便會部署最後一個已知良好的應用程式修訂版本。您可以在使用主控台建立應用程式、建立部署群組或更新部署群組時設定部署群組的選擇性設定。當您建立新的部署時，您也可以選擇覆寫先前為部署群組指定的自動轉返組態。

- 您可以在發生問題時，透過選擇其中一個或所有下列項目，讓部署轉返至最近一個已知良好的修訂：

- 部署失敗時回復。 CodeDeploy 會將最後一個已知的良好修訂版重新部署為新部署。
- Roll back when alarm thresholds are met (在到達警示閾值時轉返)。如果您在上一個步驟中新增警示至此應用程式，則 CodeDeploy 會在啟動一或多個指定的警示時重新部署最後一個已知的正確修訂版本。

> **Note**
>
> 若要暫時忽略轉返組態，請選擇 Disable rollbacks (停用轉返)。這個選擇在您希望暫時停用自動轉返，又不想要稍後再次設定相同組態時非常有用。

如需詳細資訊，請參閱 使用以下方式重新部署和復原部署 CodeDeploy。

過期執行個體的自動更新：在某些情況下， CodeDeploy 可能會將應用程式的過期修訂版部署到 Amazon EC2 執行個體。例如，如果您的 EC2 執行個體在 CodeDeploy 部署進行時啟動到 Auto Scaling 群組 (ASG)，則這些執行個體會收到應用程式的舊版本，而不是最新版本。為了使這些執行個體保持最新狀態，請 CodeDeploy 自動啟動後續部署 (在第一個執行個體之後立即)，以更新任何過時的執行個體。如果您想更改此默認行為，以便將過時的 EC2 實例保留在較舊的版本中，則可以通過 CodeDeploy API 或 AWS Command Line Interface （ CLI ）執行此操作。

若要透過 API 設定過期執行個體的自動更新，請在`UpdateDeploymentGroup`或`CreateDeploymentGroup`動作中加入 `outdatedInstancesStrategy` request 參數。如需詳細資訊，請參閱 AWS CodeDeploy API 參考資料。

若要透過配置自動更新 AWS CLI，請使用下列其中一個指令：

```
aws deploy update-deployment-group arguments --outdated-instances-strategy
UPDATE|IGNORE
```

或者...

```
aws deploy create-deployment-group arguments --outdated-instances-strategy
UPDATE|IGNORE
```

... 其中`##`會取代為部署所需的引數，而 *UPDATE|IGNORE* 會取代為啟`UPDATE`用自動更新或停`IGNORE`用它們。

範例：

```
aws deploy update-deployment-group --application-name "MyApp" --current-deployment-group-name "MyDG" --region us-east-1 --outdated-instances-strategy IGNORE
```

如需這些 AWS CLI 指令的詳細資訊，請參閱《指AWS CLI 令參考》。

使用刪除部署群組 CodeDeploy

您可以使用 CodeDeploy 主控台 AWS CLI、或 CodeDeploy API 刪除與您 AWS 帳戶關聯的部署群組。

> **Warning**
>
> 如果刪除部署群組，與該部署群組相關聯的所有詳細資料也會從中刪除 CodeDeploy。部署群組中使用的執行個體則不會變更。這個操作無法復原。

主題

刪除部署群組 (主控台)

若要使用 CodeDeploy 主控台刪除部署群組：

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在瀏覽窗格中，展開 [部署]，然後選擇 [應用程式]。
3. 在應用程式清單中，選擇與部署群組相關聯的應用程式名稱。
4. 在 Application details (應用程式詳細資訊) 頁面上，於 Deployment groups (部署群組) 標籤內，選擇您希望刪除的部署群組名稱。

5. 在 Deployment details (部署詳細資訊) 頁面上，選擇 Delete (刪除)。
6. 當系統出現提示時，請輸入部署群組的名稱，以確認要執行刪除動作，接著選擇 Delete (刪除)。

刪除部署群組 (CLI)

若要使用 AWS CLI 刪除部署群組，請呼叫命delete-deployment-group令，並指定：

- 與部署群組建立關聯的應用程式名稱。若要檢視應用程式名稱清單，請呼叫清單應用程式命令。
- 與應用程式建立關聯的部署群組名稱。若要檢視部署群組名稱清單，請呼叫命list-deployment-groups令。

使用的應用程式修訂 CodeDeploy

在中 CodeDeploy，修訂版包含 CodeDeploy 將部署到執行個體的來源檔案版本，或將在您的執行個體上執行的指令碼 CodeDeploy 。

您可以規劃修訂，將 AppSpec 檔案新增至修訂，然後將修訂推送至 Amazon S3 或 GitHub。在推送修訂後，您就可以將其部署。

主題

規劃修訂 CodeDeploy

妥善的規劃，能更輕鬆地部署修訂版。

對於 AWS Lambda 或 Amazon ECS 運算平台的部署，修訂版本與 AppSpec 檔案相同。下列資訊並不適用。如需更多資訊，請參閱將應用程式規格檔案新增至修訂 CodeDeploy

對於 EC2 /內部部署計算平台的部署，請先在開發機器上建立空的根目錄 (資料夾)。您可以在此處存放要部署至執行個體的來源檔案 (例如文字、二進位檔案、可執行檔、套件等)，或要在執行個體上執行的指令碼。

例如，在 Linux、macOS 或 Unix 的`/tmp/`根資料夾或視窗中的`c:\temp`根資料夾中：

```
/tmp/ or c:\temp (root folder)
  |--content (subfolder)
  |    |--myTextFile.txt
  |    |--mySourceFile.rb
  |    |--myExecutableFile.exe
  |    |--myInstallerFile.msi
  |    |--myPackage.rpm
  |    |--myImageFile.png
```

```
  |--scripts (subfolder)
  |    |--myShellScript.sh
  |    |--myBatchScript.bat
  |    |--myPowerShellScript.ps1
  |--appspec.yml
```

根資料夾也應包含應用程式規格檔AppSpec 案 (file)，如下所示。如需更多詳細資訊，請參閱 將應用程式規格檔案新增至修訂 CodeDeploy。

將應用程式規格檔案新增至修訂 CodeDeploy

本主題說明如何將 AppSpec 檔案加入至您的部署。它還包括用於為 AWS Lambda 和 EC2 /內部部署建立 AppSpec 檔案的範本。

主題

- 為 Amazon ECS 部署添加 AppSpec 文件
- 為 AWS Lambda 部署新增 AppSpec 檔案
- 新增EC2/ AppSpec 內部部署的檔案

為 Amazon ECS 部署添加 AppSpec 文件

對於部署到 Amazon ECS 運算平台：

- 該 AppSpec 檔案指定用於部署的 Amazon ECS 任務定義、用於路由流量的容器名稱和連接埠對應，以及在部署生命週期事件之後執行的選用 Lambda 函數。
- 修訂與 AppSpec 檔案相同。
- 一個 AppSpec 文件可以使用 JSON 或 YAML 來寫入。
- 建立 AppSpec 部署時，檔案可以儲存為文字檔或直接輸入主控台。如需詳細資訊，請參閱 建立 Amazon ECS 運算平台部署 (主控台)。

建立 AppSpec 檔案的步驟

1. 將 JSON 或 YAML 範本複製到文字編輯器或主控台的 AppSpec 編輯器中。
2. 可視需要修改範本。
3. 使用 JSON 或 YAML 驗證程式來驗證您的 AppSpec 檔案。如果使用 AppSpec編輯器，則在選擇「建立部署」時驗證檔案。

4. 如果您是使用文字編輯器，請儲存該檔案。如果您使 AWS CLI 用建立部署，請參考檔案 (如果 AppSpec 檔案位於您的硬碟或 Amazon S3 儲存貯體中)。如果您使用主控台，則必須將 AppSpec 檔案推送到 Amazon S3。

具有指示的 Amazon ECS 部署的 YAML AppSpec 檔案範本

以下是 Amazon ECS 部署 AppSpec 檔案的 YAML 範本，其中包含所有可用選項。如需在 hooks 區段中使用之生命週期事件的相關資訊，請參閱AppSpec Amazon ECS 部署的「掛鉤」部分。

```
# This is an appspec.yml template file for use with an Amazon ECS deployment in
 CodeDeploy.
# The lines in this template that start with the hashtag are
#   comments that can be safely left in the file or
#   ignored.
# For help completing this file, see the "AppSpec File Reference" in the
#   "CodeDeploy User Guide" at
#   https://docs.aws.amazon.com/codedeploy/latest/userguide/app-spec-ref.html
version: 0.0
# In the Resources section, you must specify the following: the Amazon ECS service,
 task definition name,
# and the name and port of the load balancer to route traffic,
# target version, and (optional) the current version of your AWS Lambda function.
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "" # Specify the ARN of your task definition
 (arn:aws:ecs:region:account-id:task-definition/task-definition-family-name:task-
definition-revision-number)
        LoadBalancerInfo:
          ContainerName: "" # Specify the name of your Amazon ECS application's
 container
          ContainerPort: "" # Specify the port for your container where traffic
 reroutes
# Optional properties
        PlatformVersion: "" # Specify the version of your Amazon ECS Service
        NetworkConfiguration:
          AwsvpcConfiguration:
            Subnets: ["",""] # Specify one or more comma-separated subnets in your
 Amazon ECS service
            SecurityGroups: ["",""] # Specify one or more comma-separated security
 groups in your Amazon ECS service
```

```
            AssignPublicIp: "" # Specify "ENABLED" or "DISABLED"
# (Optional) In the Hooks section, specify a validation Lambda function to run during
# a lifecycle event.
Hooks:
# Hooks for Amazon ECS deployments are:
    - BeforeInstall: "" # Specify a Lambda function name or ARN
    - AfterInstall: "" # Specify a Lambda function name or ARN
    - AfterAllowTestTraffic: "" # Specify a Lambda function name or ARN
    - BeforeAllowTraffic: "" # Specify a Lambda function name or ARN
    - AfterAllowTraffic: "" # Specify a Lambda function name or ARN
```

Amazon ECS 部署範本的 JSON AppSpec 檔案

以下是 Amazon ECS 部署 AppSpec 檔案的 JSON 範本，其中包含所有可用選項。如需範本指示，請參閱上一節中 YAML 版本的註釋。如需在 hooks 區段中使用之生命週期事件的相關資訊，請參閱AppSpec Amazon ECS 部署的「掛鉤」部分。

```
{
 "version": 0.0,
 "Resources": [
  {
   "TargetService": {
    "Type": "AWS::ECS::Service",
    "Properties": {
       "TaskDefinition": "",
       "LoadBalancerInfo": {
        "ContainerName": "",
        "ContainerPort":
       },
       "PlatformVersion": "",
       "NetworkConfiguration": {
        "AwsvpcConfiguration": {
         "Subnets": [
          "",
          ""
         ],
         "SecurityGroups": [
          "",
          ""
         ],
         "AssignPublicIp": ""
        }
       }
```

```
      }
    }
   }
  ],
  "Hooks": [
   {
    "BeforeInstall": ""
   },
   {
    "AfterInstall": ""
   },
   {
    "AfterAllowTestTraffic": ""
   },
   {
    "BeforeAllowTraffic": ""
   },
   {
    "AfterAllowTraffic": ""
   }
  ]
}
```

為 AWS Lambda 部署新增 AppSpec 檔案

若要部署至 AWS Lambda 運算平台：

- 此 AppSpec 檔案包含要部署和用於部署驗證之 Lambda 函數的相關指示。
- 修訂與 AppSpec 檔案相同。
- 一個 AppSpec 文件可以使用 JSON 或 YAML 來寫入。
- 建立部署時，AppSpec 檔案可以儲存為文字檔或直接輸入主控台 AppSpec 編輯器中。如需詳細資訊，請參閱 建立 AWS Lambda 運算平台部署 (主控台)。

若要建立 AppSpec 檔案：

1. 將 JSON 或 YAML 範本複製到文字編輯器或主控台的 AppSpec 編輯器中。
2. 可視需要修改範本。
3. 使用 JSON 或 YAML 驗證程式來驗證您的 AppSpec 檔案。如果使用 AppSpec編輯器，則在選擇「建立部署」時驗證檔案。

4. 如果您是使用文字編輯器，請儲存該檔案。如果您使 AWS CLI 用建立部署，請參考檔案 (如果 AppSpec 檔案位於您的硬碟或 Amazon S3 儲存貯體中)。如果您使用主控台，則必須將 AppSpec 檔案推送到 Amazon S3。

具有指示的 AWS Lambda 部署的 YAML AppSpec 檔案範本

如需在關聯區段中使用生命週期事件的資訊，請參閱 AppSpec AWS Lambda 部署的「掛鉤」部分。

```
# This is an appspec.yml template file for use with an AWS Lambda deployment in
 CodeDeploy.
# The lines in this template starting with the hashtag symbol are
#   instructional comments and can be safely left in the file or
#   ignored.
# For help completing this file, see the "AppSpec File Reference" in the
#   "CodeDeploy User Guide" at
#   https://docs.aws.amazon.com/codedeploy/latest/userguide/app-spec-ref.html
version: 0.0
# In the Resources section specify the name, alias,
# target version, and (optional) the current version of your AWS Lambda function.
Resources:
  - MyFunction: # Replace "MyFunction" with the name of your Lambda function
      Type: AWS::Lambda::Function
      Properties:
        Name: "" # Specify the name of your Lambda function
        Alias: "" # Specify the alias for your Lambda function
        CurrentVersion: "" # Specify the current version of your Lambda function
        TargetVersion: "" # Specify the version of your Lambda function to deploy
# (Optional) In the Hooks section, specify a validation Lambda function to run during
# a lifecycle event. Replace "LifeCycleEvent" with BeforeAllowTraffic
# or AfterAllowTraffic.
Hooks:
    - LifeCycleEvent: "" # Specify a Lambda validation function between double-quotes.
```

AWS Lambda 部署範本的 JSON AppSpec 檔案

在下列範本中，將 "MyFunction" 取代為 AWS Lambda 函數的名稱。在選用的 Hook 區段中，將生命週期事件取代為 BeforeAllowTraffic 或 AfterAllowTraffic。

如需在關聯區段中使用生命週期事件的資訊，請參閱 AppSpec AWS Lambda 部署的「掛鉤」部分。

```
{
```

```
  "version": 0.0,
  "Resources": [{
   "MyFunction": {
    "Type": "AWS::Lambda::Function",
    "Properties": {
     "Name": "",
     "Alias": "",
     "CurrentVersion": "",
     "TargetVersion": ""
    }
   }
  }],
  "Hooks": [{
    "LifeCycleEvent": ""
   }
  ]
 }
```

新增EC2/ AppSpec 內部部署的檔案

如果沒有 AppSpec 檔案， CodeDeploy 則無法將應用程式修訂版中的來源檔案對應至其目的地，或執行部署的指令碼至 EC2 /內部部署計算平台。

每個修訂只能包含一個 AppSpec 檔案。

若要將 AppSpec 檔案加入至修訂：

1. 將範本複製到文字編輯器。
2. 可視需要修改範本。
3. 使用 YAML 驗證程式來檢查檔案的 AppSpec 有效性。
4. 在修訂的根目錄中，將檔案儲存為 `appspec.yml`。
5. 執行下列其中一個指令，以確認您已將 AppSpec 檔案放置在根目錄中：

   - 若為 Linux、macOS 或 Unix：

     ```
     find /path/to/root/directory -name appspec.yml
     ```

     如果在那裡找不到 AppSpec 文件，則不會有輸出。

   - 針對 Windows：

```
dir path\to\root\directory\appspec.yml
```

如果檔案未儲存在該處，則會顯示「找不到 AppSpec 檔案」錯誤。

6. 將修訂推送到 Amazon S3 或 GitHub.

如需說明，請參閱將修訂推送 CodeDeploy 至 Amazon S3 (僅適用於 EC2 /內部部署)。

AppSpec EC2/內部部署的檔案範本與指示

> **Note**
>
> 對 Windows 伺服器執行個體的部署不支援該runas元素。如果您要部署到 Windows 伺服器執行個體，請勿將其包含在您的 AppSpec 檔案中。

```
# This is an appspec.yml template file for use with an EC2/On-Premises deployment in
 CodeDeploy.
# The lines in this template starting with the hashtag symbol are
#   instructional comments and can be safely left in the file or
#   ignored.
# For help completing this file, see the "AppSpec File Reference" in the
#   "CodeDeploy User Guide" at
#   https://docs.aws.amazon.com/codedeploy/latest/userguide/app-spec-ref.html
version: 0.0
# Specify "os: linux" if this revision targets Amazon Linux,
#   Red Hat Enterprise Linux (RHEL), or Ubuntu Server
#   instances.
# Specify "os: windows" if this revision targets Windows Server instances.
# (You cannot specify both "os: linux" and "os: windows".)
os: linux
# os: windows
# During the Install deployment lifecycle event (which occurs between the
#   BeforeInstall and AfterInstall events), copy the specified files
#   in "source" starting from the root of the revision's file bundle
#   to "destination" on the Amazon EC2 instance.
# Specify multiple "source" and "destination" pairs if you want to copy
#   from multiple sources or to multiple destinations.
# If you are not copying any files to the Amazon EC2 instance, then remove the
#   "files" section altogether. A blank or incomplete "files" section
```

```
#   may cause associated deployments to fail.
files:
  - source:
    destination:
  - source:
    destination:
# For deployments to Amazon Linux, Ubuntu Server, or RHEL instances,
#   you can specify a "permissions"
#   section here that describes special permissions to apply to the files
#   in the "files" section as they are being copied over to
#   the Amazon EC2 instance.
#   For more information, see the documentation.
# If you are deploying to Windows Server instances,
#   then remove the
#   "permissions" section altogether. A blank or incomplete "permissions"
#   section may cause associated deployments to fail.
permissions:
  - object:
    pattern:
    except:
    owner:
    group:
    mode:
    acls:
      -
    context:
      user:
      type:
      range:
    type:
      -
# If you are not running any commands on the Amazon EC2 instance, then remove
#   the "hooks" section altogether. A blank or incomplete "hooks" section
#   may cause associated deployments to fail.
hooks:
# For each deployment lifecycle event, specify multiple "location" entries
#   if you want to run multiple scripts during that event.
# You can specify "timeout" as the number of seconds to wait until failing the
 deployment
#   if the specified scripts do not run within the specified time limit for the
#   specified event. For example, 900 seconds is 15 minutes. If not specified,
#   the default is 1800 seconds (30 minutes).
#   Note that the maximum amount of time that all scripts must finish executing
#   for each individual deployment lifecycle event is 3600 seconds (1 hour).
```

```
#   Otherwise, the deployment will stop and CodeDeploy will consider the deployment
#   to have failed to the Amazon EC2 instance. Make sure that the total number of
 seconds
#   that are specified in "timeout" for all scripts in each individual deployment
#   lifecycle event does not exceed a combined 3600 seconds (1 hour).
# For deployments to Amazon Linux, Ubuntu Server, or RHEL instances,
#   you can specify "runas" in an event to
#   run as the specified user. For more information, see the documentation.
#   If you are deploying to Windows Server instances,
#   remove "runas" altogether.
# If you do not want to run any commands during a particular deployment
#   lifecycle event, remove that event declaration altogether. Blank or
#   incomplete event declarations may cause associated deployments to fail.
# During the ApplicationStop deployment lifecycle event, run the commands
#   in the script specified in "location" starting from the root of the
#   revision's file bundle.
  ApplicationStop:
    - location:
      timeout:
      runas:
    - location:
      timeout:
      runas:
# During the BeforeInstall deployment lifecycle event, run the commands
#   in the script specified in "location".
  BeforeInstall:
    - location:
      timeout:
      runas:
    - location:
      timeout:
      runas:
# During the AfterInstall deployment lifecycle event, run the commands
#   in the script specified in "location".
  AfterInstall:
    - location:
      timeout:
      runas:
    - location:
      timeout:
      runas:
# During the ApplicationStart deployment lifecycle event, run the commands
#   in the script specified in "location".
  ApplicationStart:
```

```
    - location:
      timeout:
      runas:
    - location:
      timeout:
      runas:
# During the ValidateService deployment lifecycle event, run the commands
#   in the script specified in "location".
  ValidateService:
    - location:
      timeout:
      runas:
    - location:
      timeout:
      runas:
```

選擇 CodeDeploy 存放庫類型

所需文件的存儲位置稱 CodeDeploy 為存儲庫。存放庫的使用取決於部署使用的運算平台。

- EC2/ 內部部署：若要將應用程式程式碼部署到一或多個執行個體，您的程式碼必須捆綁到封存檔案中，並放置在 CodeDeploy 可在部署程序期間存取該檔案的存放庫中。您可以將可部署的內容和 AppSpec 檔案捆綁到封存檔案中，然後將其上傳到支援的其中一個存放庫類型。 CodeDeploy
- AWS Lambda 和 Amazon ECS：部署需要一個 AppSpec 檔案，您可以透過下列其中一種方式在部署期間存取檔案：
  - 從 Amazon S3 桶。
  - 從直接輸入到控制台編 AppSpec 輯器中的文本。如需詳細資訊，請參閱 建立 AWS Lambda 運算平台部署 (主控台) 及 建立 Amazon ECS 運算平台部署 (主控台)。
  - 如果您使用 AWS CLI，您可以參考硬碟或網路磁碟機上的 AppSpec 檔案。如需詳細資訊，請參閱 建立 AWS Lambda 運算平台部署 及 建立 Amazon ECS 運算平台部署 (CLI)。

CodeDeploy 目前支援下列存放庫類型：

儲存庫類型	儲存庫詳細資訊	支援的運算平台
Amazon Simple Storage Service (Amazon S3)	Amazon 簡單儲存服務 (Amazon S3) 是安全、可擴展的物件儲存 AWS 解決方案。	使用下列運算平台的部署可將修訂版存放在 Amazon S3 儲存貯體中。

	Amazon S3 將資料當做物件存放在儲存貯體中。物件是由檔案與描述該檔案的任何選用中繼資料所組成。 若要在 Amazon S3 中存放物件，請將檔案上傳到儲存貯體。當您上傳檔案時，您可以設定物件的許可和中繼資料。 進一步了解： • 在 Amazon S3 中創建一個存儲桶 • 將修訂推送 CodeDeploy 至 Amazon S3 (僅適用於 EC2 / 內部部署) • 使用從 Amazon S3 自動部署 CodeDeploy	• EC2/內部部署 • AWS Lambda • Amazon ECS
GitHub	您可以將應用程式修訂版GitHub儲存在儲存庫中。每當 GitHub 儲存庫中的原始程式碼發生變更時，您都可以從儲存庫觸發部署。 進一步了解： • CodeDeploy 與整合 GitHub • 教學課程：用 CodeDeploy 來部署應用程式 GitHub	只有 EC2 /內部部署可以將修訂版本儲存在存放庫中 GitHub 。

Bitbucket	您可以使用 Bitbucket 管道中的CodeDeploy 管道，將程式碼部署到 EC2 執行個體的部署群組。Bitbucket 管道提供持續整合和持續部署 (CI/CD) 功能，包括 Bitbucket 部署。CodeDeploy 管道會先將成品推送到您指定的 S3 儲存貯體，然後從儲存貯體部署程式碼工件。 進一步了解： • 請參閱比特 CodeDeploy 桶的管道	只有 EC2 /內部部署可以將修訂版本儲存在存放庫中 BitBucket 。

> **Note**
>
> 部 AWS Lambda 署僅適用於 Amazon S3 儲存庫。

將修訂推送 CodeDeploy 至 Amazon S3 (僅適用於 EC2 /內部部署)

按照中所述規劃修訂規劃修訂 CodeDeploy並將 AppSpec 檔案新增至修訂後將應用程式規格檔案新增至修訂 CodeDeploy，即可將元件檔案捆綁在一起，並將修訂推送到 Amazon S3。對於部署到 Amazon EC2 執行個體，在您推送修訂後，您可 CodeDeploy 以使用將修訂從 Amazon S3 部署到執行個體。

> **Note**
>
> CodeDeploy 也可以用來部署已推送到的修訂版本 GitHub。如需詳細資訊，請參閱您的 GitHub 文件。

我們假設您已完成開始使用 CodeDeploy中的說明來設定 AWS CLI。這對於呼叫稍後描述的 push 命令特別重要。

確保你有一個 Amazon S3 存儲桶。依照建立值區中的指示操作。

如果您的部署是 Amazon EC2 執行個體，則必須建立目標 Amazon S3 儲存貯體，或與目標執行個體位於相同的區域中。例如，如果您想要將修訂部署到美國東部 (維吉尼亞北部) 區域的某些執行個體和美國西部 (奧勒岡) 區域中的某些執行個體，則在美國東部 (維吉尼亞北部) 區域中必須有一個值區，其中包含一個修訂副本的另一個值區。在這個案例中，您需要建立兩個個別的部署，一個在美國東部 (維吉尼亞北部) 區域，另一個在美國西部 (奧勒岡) 區域，即使修訂版本在區域和值區都相同。

您必須擁有上傳到 Amazon S3 儲存貯體的許可。您可以透過 Amazon S3 儲存貯體政策指定這些許可。例如，在下列 Amazon S3 儲存貯體政策中，使用萬用字元 (*) 可讓 AWS 帳戶111122223333將檔案上傳到 Amazon S3 儲存貯體中名為的任何目錄codedeploydemobucket：

```
{
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::codedeploydemobucket/*",
            "Principal": {
                "AWS": [
                    "111122223333"
                ]
            }
        }
    ]
}
```

若要檢視您的 AWS 帳戶 ID，請參閱尋找您的 AWS 帳戶 ID。

要了解如何產生和連接 Amazon S3 儲存貯體政策，請參閱儲存貯體政策範例。

呼叫push命令的使用者至少必須具有將修訂上傳到每個目標 Amazon S3 儲存貯體的許可。例如，下列政策允許使用者將修訂上傳到名為的 Amazon S3 儲存貯體中的任何位置codedeploydemobucket：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::codedeploydemobucket/*"
        }
    ]
}
```

若要了解如何建立和附加 IAM 政策，請參閱使用政策。

使用推送修訂 AWS CLI

> **Note**
>
> 該push命令將應用程序加工品和 AppSpec 文件捆綁到修訂版中。此修訂的檔案格式，是壓縮的 ZIP 檔案。此命令無法與 AWS Lambda 或 Amazon ECS 部署搭配使用，因為每個部署都需要一個 JSON 格式或 YAML 格式檔案的修訂版本。 AppSpec

呼叫 push 命令來綁定並推送部署的修訂。它的參數是：

- --application-name：(字串) 必要。要與 CodeDeploy應用程式修訂相關聯的應用程式名稱。
- --s3-location：(字串) 必要。要上傳到 Amazon S3 之應用程式修訂版位置的相關資訊。您必須指定 Amazon S3 儲存貯體和金鑰。關鍵是修訂的名稱。 CodeDeploy 在上傳之前壓縮內容。使用 s3://*your-S3-bucket-name*/*your-key.zip* 格式。
- --ignore-hidden-files 或 --no-ignore-hidden-files：(布林值) 選用。使用`--no-ignore-hidden-files`旗標 (預設值) 將隱藏檔案捆綁並上傳到 Amazon S3。使用`--ignore-hidden-files`旗標不要將隱藏的檔案捆綁並上傳到 Amazon S3。
- --source (字串) 選用。要部署的內容位置，以及要壓縮並上傳到 Amazon S3 的開發機器上的 AppSpec 檔案。該位置指定為相對於目前目錄的路徑。如果未指定相對路徑，或為路徑使用單一句點 (".")，則會使用目前目錄。
- --description (字串) 選用。用於總結應用程式修訂的評論。如果未指定，則使用預設字串「時間 AWS CLI 」UTC 上傳」，其中「time」是目前系統時間 (UTC) 表示的目前系統時間。

您可以使用 AWS CLI 為 Amazon EC2 部署推送修訂版。推送命令的範例如下所示：

在 Linux、macOS 系統或 Unix 中：

```
aws deploy push \
```

```
  --application-name WordPress_App \
  --description "This is a revision for the application WordPress_App" \
  --ignore-hidden-files \
  --s3-location s3://codedeploydemobucket/WordPressApp.zip \
  --source .
```

在 Windows 上：

```
aws deploy push --application-name WordPress_App --description "This is a revision
 for the application WordPress_App" --ignore-hidden-files --s3-location s3://
codedeploydemobucket/WordPressApp.zip --source .
```

此命令會執行下列動作：

- 將已綁定檔案與名為 WordPress_App 的應用程式建立關聯。
- 將描述連接至修訂。
- 忽略隱藏檔案。
- 為修訂 WordPressApp.zip 命名，並將其推送至名為 codedeploydemobucket 的儲存貯體。
- 將根目錄中的所有檔案綁定至修訂。

推送成功後，您可以使用 AWS CLI 或 CodeDeploy 主控台從 Amazon S3 部署修訂版本。若要使用下列項目部署此修訂 AWS CLI：

在 Linux、macOS 系統或 Unix 中：

```
aws deploy create-deployment \
  --application-name WordPress_App \
  --deployment-config-name your-deployment-config-name \
  --deployment-group-name your-deployment-group-name \
  --s3-location bucket=codedeploydemobucket,key=WordPressApp.zip,bundleType=zip
```

在 Windows 上：

```
aws deploy create-deployment --application-name WordPress_App --deployment-config-
name your-deployment-config-name --deployment-group-name your-deployment-group-name --
s3-location bucket=codedeploydemobucket,key=WordPressApp.zip,bundleType=zip
```

如需更多詳細資訊，請參閱 使用建立部署 CodeDeploy。

使用檢視應用程式修訂版 CodeDeploy

您可以使用 CodeDeploy 主控台 AWS CLI、或 CodeDeploy API 來檢視針對指定應用程式註冊到您 AWS 帳戶的所有應用程式修訂版本的詳細資料。

如需註冊修訂的詳細資訊，請參閱在 Amazon S3 中註冊應用程式修訂版 CodeDeploy。

主題

檢視應用程式修訂版本詳細資料 (

若要檢視應用程式修訂版本詳細資訊，請執行以下步驟：

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在導覽窗格中，展開 Deploy (部署) 並選擇 Applications (應用程式)。

   > **Note**
   >
   > 如果未顯示任何項目，請確定已選取正確的區域。在導覽列的區域選取器中，選擇「區域」和「端點」中列出的其中一個區域AWS 一般參考。 CodeDeploy 僅在這些地區支援。

3. 選擇具有您想檢視之修訂版的應用程式名稱。
4. 在 Application details (應用程式詳細資訊) 頁面上，選擇 Revisions (修訂版) 標籤並檢閱為該應用程式註冊的修訂版清單。選擇修訂版，然後選擇 View details (檢視詳細資訊)。

檢視應用程式修訂詳細資訊 (CLI)

若要使用 AWS CLI 檢視應用程式修訂，請呼叫get-application-revision指令或指list-application-revisions令。

> **Note**
>
> 僅 GitHub 適用於 EC2 /內部部署部署的參考。 AWS Lambda 部署的修訂不適用於 GitHub。

若要檢視單一應用程式修訂版的詳細資訊，請呼叫指定下列指get-application-revision令：

- 應用程式名稱。若要取得應用程式名稱，請呼叫清單應用程式命令。
- 對於儲存在中的修訂版 GitHub，存 GitHub 放庫名稱和參照推送至儲存庫之應用程式修訂版的提交 ID。
- 對於存放在 Amazon S3 中的修訂，Amazon S3 儲存貯體名稱包含修訂版本、上傳存檔案的名稱和檔案類型；以及可選的是存檔檔案的 Amazon S3 版本識別碼和 ETag。如果在呼叫期間指定了版本識別元、ETag 或兩者 register-application-revision，則必須在此處指定它們。

若要檢視有關多個應用程式修訂的詳細資訊，請呼叫list-application-revisions命令，指定：

- 應用程式名稱。若要取得應用程式名稱，請呼叫清單應用程式命令。
- 或者，若要僅檢視 Amazon S3 應用程式修訂的詳細資訊，請使用包含修訂版本的 Amazon S3 儲存貯體名稱。
- 或者，若要僅檢視 Amazon S3 應用程式修訂的詳細資訊，請使用前置字串，以限制搜尋 Amazon S3 應用程式修訂版本。(如果未指定， CodeDeploy 將列出所有符合的 Amazon S3 應用程式修訂。)
- (選用) 指定是否根據每個修訂是否為部署群組的目標修訂來列出修訂詳細資訊。（ 如果未指定， CodeDeploy 將列出所有匹配的修訂。 ）
- (選用) 依照欄位名稱和順序，用於對修訂詳細資料清單進行排序。（ 如果沒有指定， CodeDeploy 將以任意順序列出結果。 ）

您可以列出所有修訂，或僅列出存放在 Amazon S3 中的修訂。您無法只列出儲存在中的修訂 GitHub。

在 Amazon S3 中註冊應用程式修訂版 CodeDeploy

如果您已呼叫 push 命令將應用程式修訂推送至 Amazon S3，則不需要註冊修訂。但是，如果您透過其他方式將修訂上傳到 Amazon S3，並希望修訂顯示在 CodeDeploy 主控台或透過 AWS CLI，請依照下列步驟先註冊修訂。

如果您已將應用程式修訂推送至 GitHub 存放庫，並希望修訂顯示在 CodeDeploy 主控台或透過 AWS CLI，您也必須遵循下列步驟。

您只能使用 AWS CLI 或 CodeDeploy API 在 Amazon S3 或註冊應用程式修訂版本 GitHub。

主題

使用 CodeDeploy (CLI) 在 Amazon S3 中註冊修訂

1. 將修訂版上傳到 Amazon S3。
2. 呼叫 register-application-revision 命令，並指定：
   - 應用程式名稱。若要檢視應用程式名稱清單，請呼叫清單應用程式命令。
   - 要註冊之修訂的資訊：
     - 包含修訂版本之 Amazon S3 儲存貯體的名稱。
     - 上傳的修訂名稱和檔案類型。對於 AWS Lambda 部署，修訂版本是以 JSON 或 YAML 撰寫的 AppSpec 檔案。對於 EC2 /內部部署部署，修訂版本包含來源檔案版本，這些檔案 CodeDeploy 將部署到執行個體或 CodeDeploy 將在執行個體上執行的指令碼。

       > **Note**
       >
       > 視窗伺服器執行個體不支援 tar 和壓縮的 tar 封存檔案格式 (.tar 和 .tar.gz)。

     - (選擇性) 修訂版本的 Amazon S3 版本識別碼。（如果未指定版本標識符， CodeDeploy 將使用最新版本。）
     - (選用) 修訂的 ETag。（如果沒有指定 ETag， CodeDeploy 將跳過對象驗證。）
   - 您希望將其與修訂建立關聯的任何描述。

您可以在命令列上指定 Amazon S3 中修訂的相關資訊，並在register-application-revision呼叫時使用此語法。（`version`並且`eTag`是可選的。）

對於 EC2 /內部部署的修訂版檔案：

```
--s3-location bucket=string,key=string,bundleType=tar|tgz|
zip,version=string,eTag=string
```

對於 L AWS ambda 部署的修訂版檔案：

```
--s3-location bucket=string,key=string,bundleType=JSON|YAML,version=string,eTag=string
```

在 GitHub中註冊修訂版本 CodeDeploy (CLI)

> **Note**
>
> AWS Lambda 部署不適用於 GitHub。

1. 將修訂上傳至您的 GitHub 儲存庫。
2. 呼叫 register-application-revision 命令，並指定：
   - 應用程式名稱。若要檢視應用程式名稱清單，請呼叫清單應用程式命令。
   - 要註冊之修訂的資訊：
     - 指派給存放庫的 GitHub 使用者或群組名稱，其中包含修訂版本，後跟正斜線 (/)，後面接著存放庫名稱。
     - 遞交的 ID，此 ID 會參考儲存庫中的修訂。
   - 您希望將其與修訂建立關聯的任何描述。

中的修訂相關資訊 GitHub 可以在命令列上指定，使用下列語法做為register-application-revision呼叫的一部分：

```
--github-location repository=string,commitId=string
```

使用中的部署 CodeDeploy

在中 CodeDeploy，部署是在一個或多個執行個體上安裝內容的程序，以及程序中涉及的元件。此內容可以由代碼，Web 和配置文件，可執行文件，包，腳本等組成。 CodeDeploy 根據您指定的組態規則，部署儲存在來源儲存庫中的內容。

如果您使用 EC2 /內部部署計算平台，則同一組執行個體的兩個部署可以同時執行。

CodeDeploy 提供兩種部署類型選項：就地部署和藍/綠部署。

- 就地部署：停止部署群組中每個執行個體上的應用程式、安裝最新的應用程式修訂版本，並啟動和驗證新版本的應用程式。您可以使用負載平衡器，以便在部署期間取消註冊每個執行個體，然後在部署完成後還原至服務。只有使用 EC2 /內部部署計算平台的部署才能使用就地部署。如需就地部署的更多資訊，請參閱就地部署概述。
- 藍/綠部署：部署的行為取決於您使用的運算平台：
  - EC2 /內部部署計算平台上的藍色/綠色：部署群組 (原始環境) 中的執行個體會使用下列步驟取代為不同的一組執行個體 (取代環境)：
    - 針對替代環境佈建執行個體。
    - 最新的應用程式修訂版會安裝在取代執行個體上。
    - 應用程式測試和系統驗證等活動會發生選擇性的等待時間。
    - 取代環境中的執行個體會使用一或多個 Elastic Load Balancing 負載平衡器登錄，導致流量重新路由傳送到這些執行個體。原始環境中的執行個體會取消註冊，並可終止或繼續執行以供其他用途使用。

    > **Note**
    >
    > 如果您使用 EC2 /內部部署運算平台，請注意藍/綠部署僅適用於 Amazon EC2 執行個體。

  - AWS Lambda 或 Amazon ECS 運算平台上的藍/綠：流量會根據初期測試、線性或all-at-once部署組態以遞增方式移動。
  - 透過藍/綠部署 AWS CloudFormation：流量會從您目前的資源轉移到更新的資源，做為 AWS CloudFormation 堆疊更新的一部分。目前僅支援 ECS 藍/綠部署。

  如需藍/綠部署的詳細資訊，請參閱 藍/綠部署概述。

如需從 Amazon S3 自動部署的相關資訊，請參閱使用從 Amazon S3 自動部署 CodeDeploy。

主題

使用建立部署 CodeDeploy

您可以使用 CodeDeploy 主控台 AWS CLI、或 CodeDeploy API 建立部署，以便在部署群組中的執行個體上安裝已推送到 Amazon S3 的應用程式修訂版，或在部署群組中的執行個體上安裝應用程式修訂版。 GitHub

建立部署的程序取決於部署所使用的運算平台。

主題

部署先決條

請先確定下列步驟完成，再開始部署。

AWS Lambda 計算平台上的部署必要條件

- 建立包含至少一個部署群組的應用程式。如需詳細資訊，請參閱 建立應用程式 CodeDeploy 及 建立部署群組 CodeDeploy。
- 準備指定您要部署的 Lambda 函數版本的應用程式修訂版本 (也稱為 AppSpec 檔案)。該 AppSpec 文件還可以指定 Lambda 函數來驗證您的部署。如需更多資訊，請參閱使用的應用程式修訂 CodeDeploy。
- 如果您想要使用部署的自訂部署組態，請於開始部署程序之前予以建立。如需相關資訊，請參閱Create a Deployment Configuration。

EC2/內部部署計算平台上的部署必備條件

- 針對就地部署，建立或設定您要在其中部署的執行個體。如需相關資訊，請參閱使用的例證 CodeDeploy。對於藍/綠部署，您可以擁有一個現有的 Amazon EC2 Auto Scaling 群組作為替代環境的範本，或者您有一個或多個執行個體或指定為原始環境的 Amazon EC2 Auto Scaling 群組。如需詳細資訊，請參閱 教學課程：用 CodeDeploy 於將應用程式部署到 Auto Scaling 群組 及 CodeDeploy 與 Amazon EC2 Auto Scaling 集成。
- 建立包含至少一個部署群組的應用程式。如需詳細資訊，請參閱 建立應用程式 CodeDeploy 及 建立部署群組 CodeDeploy。
- 準備您要部署至部署群組中執行個體的應用程式修訂。如需相關資訊，請參閱使用的應用程式修訂 CodeDeploy。
- 如果您想要使用部署的自訂部署組態，請於開始部署程序之前予以建立。如需相關資訊，請參閱Create a Deployment Configuration。
- 如果您要從 Amazon S3 儲存貯體部署應用程式修訂版，則儲存貯體與部署群組中的執行個體位於相同的 AWS 區域。
- 如果您要從 Amazon S3 儲存貯體部署應用程式修訂版，則 Amazon S3 儲存貯體政策已套用至儲存貯體。此政策會將下載應用程式修訂所需的許可授予您的執行個體。

  例如，下列 Amazon S3 儲存貯體政策允許任何具有附加 IAM 執行個體設定檔 (包含 ARN`arn:aws:iam::444455556666:role/CodeDeployDemo`) 的 Amazon EC2 執行個體從 Amazon S3 儲存貯體的任何位置下載名為`codedeploydemobucket`：

```
{
    "Statement": [
        {
            "Action": [
```

```
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::codedeploydemobucket/*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666:role/CodeDeployDemo"
                ]
            }
        }
    ]
}
```

下列 Amazon S3 儲存貯體政策允許任何具codedeploydemobucket有相關聯 IAM 使用者 (包含 ARNarn:aws:iam::444455556666:user/CodeDeployUser) 的現場部署執行個體從 Amazon S3 儲存貯體中的任何位置下載：

```
{
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::codedeploydemobucket/*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666:user/CodeDeployUser"
                ]
            }
        }
    ]
}
```

如需如何產生和連接 Amazon S3 儲存貯體政策的詳細資訊，請參閱儲存貯體政策範例。

- 如果您要建立藍色/綠色部署，或已在部署群組中為就地部署指定選用的 Classic Load Balancer、應用程式負載平衡器或 Network Load Balancer，則您已使用 Amazon VPC 建立了至少包含兩個子網路的 VPC。(CodeDeploy 使用 Elastic Load Balancing，這要求負載平衡器群組中的所有執行個體都位於單一 VPC 中。)

如果您尚未建立 VPC，請參閱 Amazon VPC 入門指南。

- 如果您要建立藍/綠部署，則至少已在 Elastic Load Balancing 中設定一個 Classic Load Balancer、應用程式負載平衡器或 Network Load Balancer，並使用它來註冊組成您原始環境的執行個體。

> **Note**
>
> 稍後會向負載平衡器註冊替換環境中的執行個體。

如需有關設定負載平衡器的詳細資訊在 Elastic Load Balancing 中為 CodeDeploy Amazon EC2 部署設定負載平衡器，請參閱和為 CodeDeploy Amazon ECS 部署設定負載平衡器、目標群組和接聽程式。

藍色/綠色部署的部署必要條件 AWS CloudFormation

- 您的範本不需要為 CodeDeploy 應用程式或部署群組建立資源模型。
- 您的範本必須包含使用至少包含兩個子網路的 Amazon VPC 的 VPC 資源。
- 您的範本必須包含 Elastic Load Balancing 中一或多個傳統負載平衡器、應用程式負載平衡器或網路負載平衡器的資源，這些資源用於將流量導向目標群組。

建立 Amazon ECS 運算平台部署 (主控台)

本主題說明如何使用主控台部署 Amazon ECS 服務。如需詳細資訊，請參閱 教學課程：將應用程式部署到 Amazon ECS 及 教學課程：透過驗證測試部署 Amazon ECS 服務。

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 執行以下任意一項：

   - 如果您要部署應用程式，請在導覽窗格中展開 Deploy (部署)，然後選擇 Applications (應用程式)。選擇您要部署的應用程式名稱。請確定您應用程式的運算平台欄位是 Amazon ECS。

- 如果您要重新部署一個部署，請在導覽窗格中展開 Deploy (部署)，然後選擇 Deployments (部署)。選擇您要重新部署的部署，然後在 Application (應用程式) 欄中選擇其應用程式的名稱。請確定您部署的運算平台欄位是 Amazon ECS。

3. 在 Deployments (部署) 標籤上，選擇 Create deployment (建立部署)。

> **Note**
>
> 您的應用程式必須先有部署群組，才能部署它。如果您的應用程式沒有部署群組，請在 [部署群組] 索引標籤上選擇 [建立部署群組]。如需詳細資訊，請參閱 建立部署群組 CodeDeploy。

4. 在 Deployment group (部署群組) 中，選擇要用於此部署的部署群組。
5. 在 Revision location (修訂版本位置) 旁，選擇修訂版本的所在位置：

   - 我的應用程式存放在 Amazon S3 — 如需詳細資訊指定存放在 Amazon S3 儲存貯體中之修訂版本的相關資訊，請參閱，然後返回步驟 6。
   - 使用 AppSpec 編輯器 — 選取 [JSON] 或 [YAML]，然後在編輯器中輸入您的 AppSpec 檔案。您可以選擇另存為文字 AppSpec 檔案來儲存檔案。當您在這些步驟結束時選擇 Deploy (部署) 時，如果您的 JSON 或 YAML 無效，則會收到錯誤。如需建立 AppSpec 檔案的更多資訊，請參閱將應用程式規格檔案新增至修訂 CodeDeploy。

6. (選用) 在 Deployment description (部署描述) 方塊中，輸入此部署的描述。
7. (選用) 在 Rollback configuration overrides (轉返組態覆寫) 中，您可以指定此部署的自動轉返選項，而其與部署群組所指定的選項 (如果有的話) 不同。

   若要取得有關中復原的資訊 CodeDeploy，請參閱重新部署和部署復原和使用以下方式重新部署和復原部署 CodeDeploy。

   請選擇下列項目：

   - 部署失敗時復原 — CodeDeploy 將最後一個已知的良好修訂版重新部署為新部署。
   - 符合警示閾值時復原 — 如果已 CodeDeploy 將警示新增至部署群組，則會在啟動一或多個指定警示時重新部署上次已知的正確修訂版本。
   - 停用復原 — 不執行此部署的復原。

8. 選擇 Create deployment (建立部署)。

   如需追蹤部署的狀態，請參閱檢視 CodeDeploy 部署詳情 。

建立 AWS Lambda 運算平台部署 (主控台)

本主題說明如何使用主控台部署 Lambda 函數。

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 執行以下任意一項：

   - 如果您要部署應用程式，請在導覽窗格中展開 Deploy (部署)，然後選擇 Applications (應用程式)。選擇您要部署的應用程式名稱。請確定應用程式的運算平台資料行是 AWS Lambda。
   - 如果您要重新部署一個部署，請在導覽窗格中展開 Deploy (部署)，然後選擇 Deployments (部署)。選擇您要重新部署的部署，然後在 Application (應用程式) 欄中選擇其應用程式的名稱。請確定您部署的運算平台資料行是 AWS Lambda。

3. 在 Deployments (部署) 標籤上，選擇 Create deployment (建立部署)。

   > **Note**
   >
   > 您的應用程式必須先有部署群組，才能部署它。如果您的應用程式沒有部署群組，請在 [部署群組] 索引標籤上選擇 [建立部署群組]。如需詳細資訊，請參閱 建立部署群組 CodeDeploy。

4. 在 Deployment group (部署群組) 中，選擇要用於此部署的部署群組。
5. 在 Revision location (修訂版本位置) 旁，選擇修訂版本的所在位置：

   - 我的應用程式存放在 Amazon S3 — 如需詳細資訊指定存放在 Amazon S3 儲存貯體中之修訂版本的相關資訊，請參閱，然後返回步驟 6。
   - 使用 AppSpec 編輯器 — 選取 [JSON] 或 [YAML]，然後在編輯器中輸入您的 AppSpec 檔案。您可以選擇另存為文字 AppSpec 檔案來儲存檔案。當您在這些步驟結束時選擇 Deploy (部署) 時，如果您的 JSON 或 YAML 無效，則會收到錯誤。如需建立 AppSpec 檔案的更多資訊，請參閱將應用程式規格檔案新增至修訂 CodeDeploy。

6. (選用) 在 Deployment description (部署描述) 方塊中，輸入此部署的描述。

7. (選擇性) 展開部署群組覆寫以選擇部署組態，以控制流量轉移至與部署群組中指定的 Lambda 函數版本不同的方式。

   如需詳細資訊，請參閱 AWS Lambda 運算平台上的部署組態。

8. (選用) 在 Rollback configuration overrides (轉返組態覆寫) 中，您可以指定此部署的自動轉返選項，而其與部署群組所指定的選項 (如果有的話) 不同。

   若要取得有關中復原的資訊 CodeDeploy，請參閱重新部署和部署復原和使用以下方式重新部署和復原部署 CodeDeploy。

   請選擇下列項目：

   - 部署失敗時復原 — CodeDeploy 將最後一個已知的良好修訂版重新部署為新部署。
   - 符合警示閾值時復原 — 如果已 CodeDeploy 將警示新增至部署群組，則會在啟動一或多個指定警示時重新部署上次已知的正確修訂版本。
   - 停用復原 — 不執行此部署的復原。

9. 選擇 Create deployment (建立部署)。

   如需追蹤部署的狀態，請參閱檢視 CodeDeploy 部署詳情 。

建立 EC2/內部部署計算平台部署 (主控台)

本主題說明如何使用主控台將應用程式部署到 Amazon EC2 或現場部署伺服器。

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 執行以下任意一項：

   - 如果您要部署應用程式，請在導覽窗格中展開 Deploy (部署)，然後選擇 Applications (應用程式)。選擇您要部署的應用程式名稱。請確定應用程式的運算平台欄位是 EC2/內部部署。
   - 如果您要重新部署一個部署，請在導覽窗格中展開 Deploy (部署)，然後選擇 Deployments (部署)。找到您要重新部署的部署，然後在 Application (應用程式) 欄中選擇其應用程式的名稱。請確定您部署的「計算平台」欄位是 EC2/內部部署。

3. 在 Deployments (部署) 標籤上，選擇 Create deployment (建立部署)。

> **Note**
>
> 您的應用程式必須先有部署群組，才能部署它。如果您的應用程式沒有部署群組，請在 [部署群組] 索引標籤上選擇 [建立部署群組]。如需詳細資訊，請參閱 建立部署群組 CodeDeploy。

4. 在 Deployment group (部署群組) 中，選擇要用於此部署的部署群組。
5. 在 Repository type (儲存庫類型) 旁，選擇存放修訂版本的儲存庫類型：

   - 我的應用程式存放在 Amazon S3 — 如需詳細資訊指定存放在 Amazon S3 儲存貯體中之修訂版本的相關資訊，請參閱，然後返回步驟 6。
   - 我的應用程式儲存在 GitHub-如需詳細資訊指定儲存在存 GitHub 放庫中之修訂版本的相關資訊，請參閱，然後返回步驟 6。

6. (選用) 在 Deployment description (部署描述) 方塊中，輸入此部署的描述。
7. (選擇性) 展開覆寫部署組態以選擇部署組態，以控制流量轉移至與部署群組中指定不同之 Amazon EC2 或現場部署伺服器的方式。

   如需詳細資訊，請參閱 使用中的部署組態 CodeDeploy。

8. a. 如果您希望在 ApplicationStop生命週期事件失敗的情況下，對執行個體的部署成功，請選取 [如果**`ApplicationStop`**生命週期事件失敗，則不要讓部署失敗]。

   b. 展開其他部署行為設定，以指定如何 CodeDeploy 處理部署目標位置中不屬於先前成功部署的檔案。

      請選擇下列項目：

      - 部署失敗 — 報告錯誤且部署狀態變更為`Failed`。
      - 覆寫內容 — 如果目標位置中存在相同名稱的檔案，則應用程式修訂版本中的版本會取代它。
      - 保留內容 — 如果目標位置中存在相同名稱的檔案，則會保留該檔案，且不會將應用程式修訂版本中的版本複製到實例中。

      如需詳細資訊，請參閱 復原現有內容的行為。

9. (選用) 在 Rollback configuration overrides (轉返組態覆寫) 中，您可以指定此部署的自動轉返選項，而其與部署群組所指定的選項 (如果有的話) 不同。

若要取得有關中復原的資訊 CodeDeploy，請參閱重新部署和部署復原和使用以下方式重新部署和復原部署 CodeDeploy。

請選擇下列項目：

- 部署失敗時復原 — CodeDeploy 將最後一個已知的良好修訂版重新部署為新部署。
- 符合警示閾值時復原 — 如果已 CodeDeploy 將警示新增至部署群組，則會在啟動一或多個指定警示時部署上次已知的正確修訂版本。
- 停用復原 — 不執行此部署的復原。

10. 選擇 Start deployment (啟動部署)。

    如需追蹤部署的狀態，請參閱檢視 CodeDeploy 部署詳情 。

主題

- 指定存放在 Amazon S3 儲存貯體中之修訂版本的相關資訊
- 指定儲存在存 GitHub 放庫中之修訂版本的相關資訊

指定存放在 Amazon S3 儲存貯體中之修訂版本的相關資訊

如果您按照中的步驟進行操作建立 EC2/內部部署計算平台部署 (主控台)，請按照以下步驟新增存放在 Amazon S3 儲存貯體中的應用程式修訂版本的詳細資訊。

1. 將修訂版本的 Amazon S3 連結複製到修訂版位置。尋找連結值：

   a. 在單獨的瀏覽器標籤中：

      登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。

      瀏覽並選擇修訂。

   b. 如果未顯示 Properties (屬性) 窗格，請選擇 Properties (屬性) 按鈕。

   c. 在「屬性」窗格中，將「連結」欄位的值複製到 CodeDeploy 主控台的「修訂版位置」方塊中。

   將 ETag (檔案檢查總和) 指定為修訂位置的一部分：

- 如果 Link (連結) 欄位值的結尾為 **?versionId=*versionId***，請將 **&etag=** 和 ETag 新增至 Link (連結) 欄位值的結尾。
- 如果 Link (連結) 欄位值未指定版本 ID，請將 **?etag=** 和 ETag 新增至 Link (連結) 欄位值的結尾。

> **Note**
>
> 雖然不容易複製 Link (連結) 欄位的值，但是您也可以使用下列任一格式輸入修訂位置：
> **s3://*bucket-name*/*folders*/*objectName***
> **s3://*bucket-name*/*folders*/*objectName*?versionId=*versionId***
> **s3://*bucket-name*/*folders*/*objectName*?etag=*etag***
> **s3://*bucket-name*/*folders*/*objectName*?versionId=*versionId*&etag=*etag***
> ***bucket-name*.s3.amazonaws.com/*folders*/*objectName***

2. 如果訊息顯示在 File type (檔案類型) 清單中指出偵測不到檔案類型，請選擇修訂的檔案類型。否則，請接受偵測到的檔案類型。

指定儲存在存 GitHub 放庫中之修訂版本的相關資訊

如果您遵循中的步驟建立 EC2/內部部署計算平台部署 (主控台)，請遵循下列步驟來新增儲存在存 GitHub 放庫中之應用程式修訂版本的詳細資訊。

1. 在 Connect 至中 GitHub，執行下列其中一個動作：
   - 若要為 CodeDeploy 應用程式與 GitHub 帳戶建立連線，請在其他網頁瀏覽器索引標籤中登出 GitHub。在GitHub 帳戶中，輸入識別此連線的名稱，然後選擇 [Connect 線至] GitHub。該網頁會提示您授權 CodeDeploy 與 GitHub 您的應用程序進行交互。繼續步驟 2。
   - 若要使用已建立的連線，請在GitHub帳戶中選取其名稱，然後選擇 [Connect 線至] GitHub。繼續步驟 4。
   - 若要建立與其他 GitHub 帳戶的連線，請在不同的網頁瀏覽器索引標籤中登出 GitHub。選擇 [Connect 到其他 GitHub 帳戶]，然後選擇 [Connect 到] GitHub。繼續步驟 2。
2. 如果系統提示您登入 GitHub，請依照「登入」頁面上的指示進行。 GitHub 使用您的使用者名稱或電子郵件和密碼登入。
3. 如果出現 Authorize application (授權應用程式) 頁面，請選擇 Authorize application (授權應用程式)。

4. 在 [建立部署] 頁面的 [存放庫名稱] 方塊中，輸入包含修訂版本的 GitHub 使用者或組織名稱，後面接著正斜線 (/)，後面接著包含修訂版本的存放庫名稱。如果您不確定數值的類型：

   a. 在其他 Web 瀏覽器標籤中，前往GitHub儀表板。

   b. 在 Your repositories (您的儲存庫) 中，將滑鼠指標移至目標儲存庫名稱上。工具提示隨即出現，其中顯示 GitHub 使用者或組織名稱，後面接著正斜線 (/)，後面接著存放庫的名稱。將這個顯示的數值輸入到 Repository name (儲存庫名稱) 方塊中。

   > **Note**
   >
   > 如果您的儲存庫中看不到目標存放庫名稱，請使用 GitHub「搜尋」方塊來尋找目標存放庫名稱以及使用 GitHub 者或組織名稱。

5. 在 Commit ID (遞交 ID) 中，輸入遞交的 ID，其可參考儲存庫中的修訂。如果您不確定數值的類型：

   a. 在其他 Web 瀏覽器標籤中，前往GitHub儀表板。

   b. 在 Your repositories (您的儲存庫)中，請選擇包含目標遞交的儲存庫。

   c. 在遞交列表中，尋找及複製遞交 ID，其可參考儲存庫中的修訂。此 ID 通常長度為 40 個字元，並且由字母和數字所組成。(請勿使用短版的遞交 ID，其通常為長版遞交 ID 的前 10 個字元)。

   d. 將遞交 ID 貼至 Commit ID (遞交 ID) 方塊中。

建立 Amazon ECS 運算平台部署 (CLI)

建立應用程式和修訂版之後 (在 Amazon ECS 部署中，這是 AppSpec 檔案)：

呼叫建立部署命令，指定：

- 應用程式名稱。若要檢視應用程式名稱清單，請呼叫清單應用程式命令。
- 部署群組名稱。若要檢視部署群組名稱清單，請呼叫命list-deployment-groups令。
- 待部署的修訂版之資訊：

  對於存放在 Amazon S3 中的修訂：

  - 包含修訂版本的 Amazon S3 儲存貯體名稱。
  - 上傳的修改版名稱。

- (選擇性) 修訂版本的 Amazon S3 版本識別碼。(如果未指定版本識別碼， CodeDeploy 會使用最新版本。)
- (選用) 修訂版的 ETag。（ 如果未指定 ETag，則 CodeDeploy 跳過對象驗證。 ）

對於存放在不在 Amazon S3 中的檔案中的修訂，您需要檔案名稱及其路徑。因為您的修訂版檔案使用 JSON 或 YAML 寫入，所以它最可能擴展 .json 或 .yaml。

- (選用) 部署描述。

修訂版檔案可以指定為上傳到 Amazon S3 儲存貯體的檔案，或以字串形式指定。作為 create-deployment 命令的一部分時的語法為：

- Amazon S3 桶：

  `version` 和 `eTag` 是選擇性使用的。

```
--s3-location bucket=string,key=string,bundleType=JSON|
YAML,version=string,eTag=string
```

- 字串：

```
--revision '{"revisionType": "String", "string": {"content":"revision-as-string"}}'
```

> **Note**
>
> create-deployment 命令可以從檔案載入修訂版。如需詳細資訊，請參閱從檔案載入參數。

如需 AWS Lambda 部署修訂範本，請參閱為 AWS Lambda 部署新增 AppSpec 檔案。如需範例修訂，請參閱 AppSpec AWS Lambda 部署的檔案範例 。

如需追蹤部署的狀態，請參閱檢視 CodeDeploy 部署詳情 。

建立 AWS Lambda 運算平台部署

建立應用程式和修訂版之後 (在 AWS Lambda 部署中，這是 AppSpec 檔案)：

呼叫建立部署命令，指定：

- 應用程式名稱。若要檢視應用程式名稱清單，請呼叫清單應用程式命令。
- 部署群組名稱。若要檢視部署群組名稱清單，請呼叫命list-deployment-groups令。
- 待部署的修訂版之資訊：

  對於存放在 Amazon S3 中的修訂：

  - 包含修訂版本的 Amazon S3 儲存貯體名稱。
  - 上傳的修改版名稱。
  - (選擇性) 修訂版本的 Amazon S3 版本識別碼。(如果未指定版本識別碼， CodeDeploy 會使用最新版本。)
  - (選用) 修訂版的 ETag。（ 如果未指定 ETag，則 CodeDeploy 跳過對象驗證。 ）

  對於存放在不在 Amazon S3 中的檔案中的修訂，您需要檔案名稱及其路徑。因為您的修訂版檔案使用 JSON 或 YAML 寫入，所以它最可能擴展 .json 或 .yaml。
- (選用) 使用的部署組態名稱。若要檢視部署規劃清單，請呼叫指list-deployment-configs令。(如果未指定，則 CodeDeploy 會使用特定的預設部署規劃。)
- (選用) 部署描述。

修訂版檔案可以指定為上傳到 Amazon S3 儲存貯體的檔案，或以字串形式指定。作為 create-deployment 命令的一部分時的語法為：

- Amazon S3 桶：

  `version` 和 `eTag` 是選擇性使用的。

```
--s3-location bucket=string,key=string,bundleType=JSON|
YAML,version=string,eTag=string
```

- 字串：

```
--revision '{"revisionType": "String", "string": {"content":"revision-as-string"}}'
```

> **Note**
>
> create-deployment 命令可以從檔案載入修訂版。如需詳細資訊，請參閱從檔案載入參數。

如需 AWS Lambda 部署修訂範本，請參閱為 AWS Lambda 部署新增 AppSpec 檔案。如需範例修訂，請參閱 AppSpec AWS Lambda 部署的檔案範例 。

如需追蹤部署的狀態，請參閱檢視 CodeDeploy 部署詳情 。

建立EC2/內部部署計算平台部署 (CLI)

若要使用將修訂部署 AWS CLI 至 EC2 /內部部署計算平台：

1. 當您將執行個體準備完成以後，建立應用程式以及發布修訂版，然後請執行以下其中一項：
   - 如果您想要從 Amazon S3 儲存貯體部署修訂版，請立即繼續執行步驟 2。
   - 如果您要從 GitHub 存放庫部署修訂版本，請先完成中的步驟將 CodeDeploy 應用程式 Connect 到儲 GitHub 存庫，然後繼續執行步驟 2。
2. 呼叫建立部署命令，指定：
   - `--application-name`：應用程式名稱。若要檢視應用程式名稱清單，請呼叫清單應用程式命令。
   - `--deployment-group-name`：Amazon EC2 部署群組名稱。若要檢視部署群組名稱清單，請呼叫命list-deployment-groups令。
   - `--revision`：要建置之修訂的相關資訊：

     對於存放在 Amazon S3 中的修訂：
     - `s3Location`：包含修訂版本的 Amazon S3 儲存貯體名稱。
     - `s3Location --> key`：已上傳修訂的名稱。
     - `s3Location --> bundleType`：上傳修訂的文件類型。

       > **Note**
       >
       > 視窗伺服器執行個體不支援 tar 和壓縮的 tar 封存檔案格式 (.tar 和 .tar.gz)。

     - `s3Location --> version`: (選用) 修訂版本的 Amazon S3 版本識別碼。(如果未指定版本識別碼， CodeDeploy 會使用最新版本。)
     - `s3Location --> eTag`: (選擇性) 修訂的 ETag。 (如果未指定 ETag，則 CodeDeploy 跳過對象驗證。)

     對於儲存在中的修訂 GitHub：
     - `gitHubLocation --> repository`：指派給包含修訂版本的儲存庫的 GitHub 使用者或群組名稱，後跟正斜線 (/)，後面接著儲存庫名稱。

  - `gitHubLocation --> commitId`：修訂的提交 ID。
- `--deployment-config-name`: (選擇性) 要使用的部署規劃名稱。若要檢視部署規劃清單，請呼叫指list-deployment-configs令。(如果未指定，則 CodeDeploy 會使用特定的預設部署規劃。)
- `--ignore-application-stop-failures | --no-ignore-application-stop-failures`: (選擇性) 如果部署生命週期事件失敗，您是否希望BeforeInstall部署至執行個體繼續執行ApplicationStop部署生命週期事件。
- `--description`: (選擇性) 部署的說明。
- `--file-exists-behavior`: (選擇性) 作為部署程序的一部分， CodeDeploy 代理程式會從每個執行個體中移除最近部署所安裝的所有檔案。選擇當不屬於先前部署的檔案出現在目標部署位置時，會發生什麼情況。
- `--target-instances`：對於藍/綠部署，在藍/綠部署中屬於替換環境的執行個體相關資訊，包括一或多個 Amazon EC2 Auto Scaling 群組的名稱，或用於識別 Amazon EC2 執行個體的標籤篩選器金鑰、類型和值。

> **Note**
>
> 使用此語法做為create-deployment呼叫的一部分，直接在命令列上指定 Amazon S3 中修訂的相關資訊。(`version` 和 `eTag` 是選擇性使用的)。
>
> ```
> --s3-location bucket=string,key=string,bundleType=tar|tgz|
> zip,version=string,eTag=string
> ```
>
> 使用此語法做為create-deployment呼叫的一部分， GitHub 直接在命令列中指定修訂的相關資訊：
>
> ```
> --github-location repository=string,commitId=string
> ```
>
> 若要取得已推送修訂的相關資訊，請呼叫指list-application-revisions令。

如需追蹤部署的狀態，請參閱檢視 CodeDeploy 部署詳情 。

建立部署命令參考

下面是命令的命令結構和選項。`create-deployment`如需詳細資訊，請參閱《命令參考》中的建立部署參考。AWS CLI

```
create-deployment
--application-name <value>
[--deployment-group-name <value>]
[--revision <value>]
[--deployment-config-name <value>]
[--description <value>]
[--ignore-application-stop-failures | --no-ignore-application-stop-failures]
[--target-instances <value>]
[--auto-rollback-configuration <value>]
[--update-outdated-instances-only | --no-update-outdated-instances-only]
[--file-exists-behavior <value>]
[--s3-location <value>]
[--github-location <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton <value>]
```

將 CodeDeploy 應用程式 Connect 到儲 GitHub 存庫

在您第一次使用 GitHub 存放庫部署應用程式之前 AWS CLI，您必須先授與 GitHub 代表您的 GitHub 帳戶進行互動的 CodeDeploy 權限。對於使用 CodeDeploy 控制台的每個應用程序，此步驟必須完成一次。

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 選擇 Applications (應用程式)。
3. 從應用程式中，選擇您要連結至 GitHub 使用者帳戶的應用程式，然後選擇部署應用程式。

   > **Note**
   >
   > 您不是在建立部署。這是目前 CodeDeploy 允許代表您的 GitHub使用者帳戶進 GitHub 行互動的唯一方法。

4. 在 [存放庫類型] 旁邊，選擇 [我的應用程式修訂儲存於] GitHub。
5. 選擇「Connect 至」 GitHub。

> **Note**
>
> 如果您看到「Connect 到其他 GitHub帳戶」連結：
> 您可能已經授權 CodeDeploy 代表應用程式的其他 GitHub 帳戶進行互動。 GitHub
> 對於連結 CodeDeploy 至中的所有應用程式，您可能已撤銷代表已登入 GitHub 帳戶與
> GitHub之互動的授權。 CodeDeploy
> 如需詳細資訊，請參閱 GitHub 使用應用程式的驗證 CodeDeploy。

6. 如果您尚未登入 GitHub，請依照「登入」頁面上的指示進行。
7. 在 授權應用程式 頁面上，請選擇 授權應用程式。
8. 現在 CodeDeploy 已取得權限，請選擇「取消」，然後繼續執行中的步驟建立EC2/內部部署計算平台部署 (CLI)。

透過以下方式建立亞馬遜ECS 藍色/綠色部署 AWS CloudFormation

您可以透過管 AWS CloudFormation 理 Amazon ECS 藍色/綠色部署。 CodeDeploy您可以透過定義綠色和藍色資源，並指定要在 AWS CloudFormation中使用的流量路由和穩定設定來產生部署。本主題涵蓋由管理的 Amazon ECS 藍/綠部署 CodeDeploy 與受管理的部署之間的差異。 AWS CloudFormation

如需使用 AWS CloudFormation 管理 Amazon ECS 藍/綠部署的相關 step-by-step 說明，請參閱使用者指南 AWS CloudFormation中的 CodeDeploy 使用自動化 ECS 藍/綠部署。AWS CloudFormation

> **Note**
>
> 亞太區域 (大阪) 地區無 AWS CloudFormation 法使用管理 Amazon ECS 藍/綠部署。

透過和的Amazon ECS 藍色/綠色部署之間的差異 CodeDeploy AWS CloudFormation

AWS CloudFormation 堆疊範本會為 Amazon ECS 任務相關資源和基礎設施建模，以及部署的設定選項。因此，標準 Amazon ECS 藍/綠部署和透過建立的藍/綠部署之間存在差異。 AWS CloudFormation

與標準 Amazon ECS 藍/綠部署不同，您不需要建立模型或手動建立下列項目：

- 您不會藉由指定唯一代表您要部署的項目的名稱來建立 AWS CodeDeploy 應用程式。

- 您不會建立 AWS CodeDeploy 部署群組。
- 您不指定應用程式規格檔案 (AppSpec 檔案)。通常使用 AppSpec 檔案管理的資訊 (例如加權組態選項或生命週期事件) 由`AWS::CodeDeploy::BlueGreen`掛接管理。

此表格摘要列出部署類型之間高階工作流程中的差異。

函式	標準藍/綠部署	藍/綠部署 AWS CloudFormation
指定 Amazon ECS 叢集、Amazon ECS 服務、應用程式負載平衡器或 Network Load Balancer、生產接聽程式、測試接聽程式和兩個目標群組。	建立指定這些資源的 CodeDeploy 部署群組。	建立 AWS CloudFormation 範本以建立這些資源的模型。
指定要部署的變更。	建立 CodeDeploy 應用程式。	建立指定容器映像檔的 AWS CloudFormation 範本。
指定 Amazon ECS 任務定義、容器名稱和容器連接埠。	建立指定這些資源的 AppSpec 檔案。	建立 AWS CloudFormation 範本以建立這些資源的模型。
指定部署流量轉移選項和生命週期事件勾點。	建立指定這些選項的 AppSpec 檔案。	建 AWS CloudFormation 立使用`AWS::CodeDeploy::BlueGreen` 勾點參數指定這些選項的範本。
CloudWatch 警報。	建立觸發回復的 CloudWatch 警示。	在 AWS CloudFormation 堆疊層級設定會觸發復原的 CloudWatch 警示。
轉返/重新部署。	指定轉返和重新部署選項。	在中取消堆疊更新 AWS CloudFormation。

監控 Amazon ECS 藍/綠部署 AWS CloudFormation

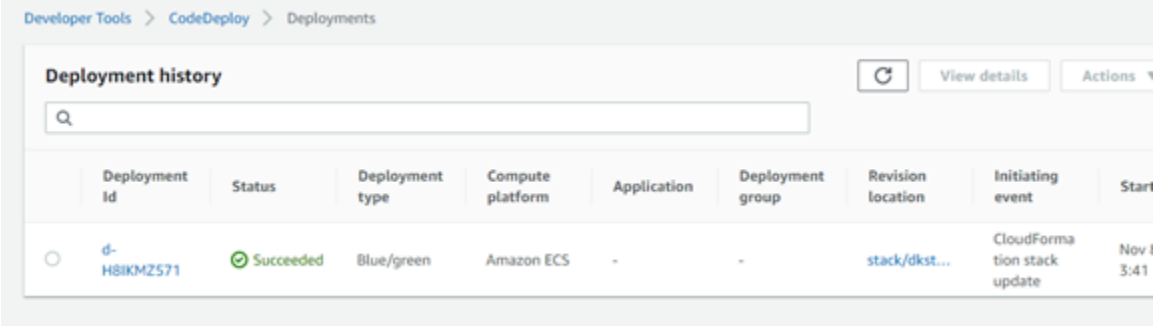
您可以透過 AWS CloudFormation 和 CodeDeploy監控藍/綠部署。若要取得監視的相關資訊 AWS CloudFormation，請參閱《AWS CloudFormation 使用指南》AWS CloudFormation中的〈監控藍/綠事件〉。

若要檢視藍色/綠色部署的部署狀態 CodeDeploy

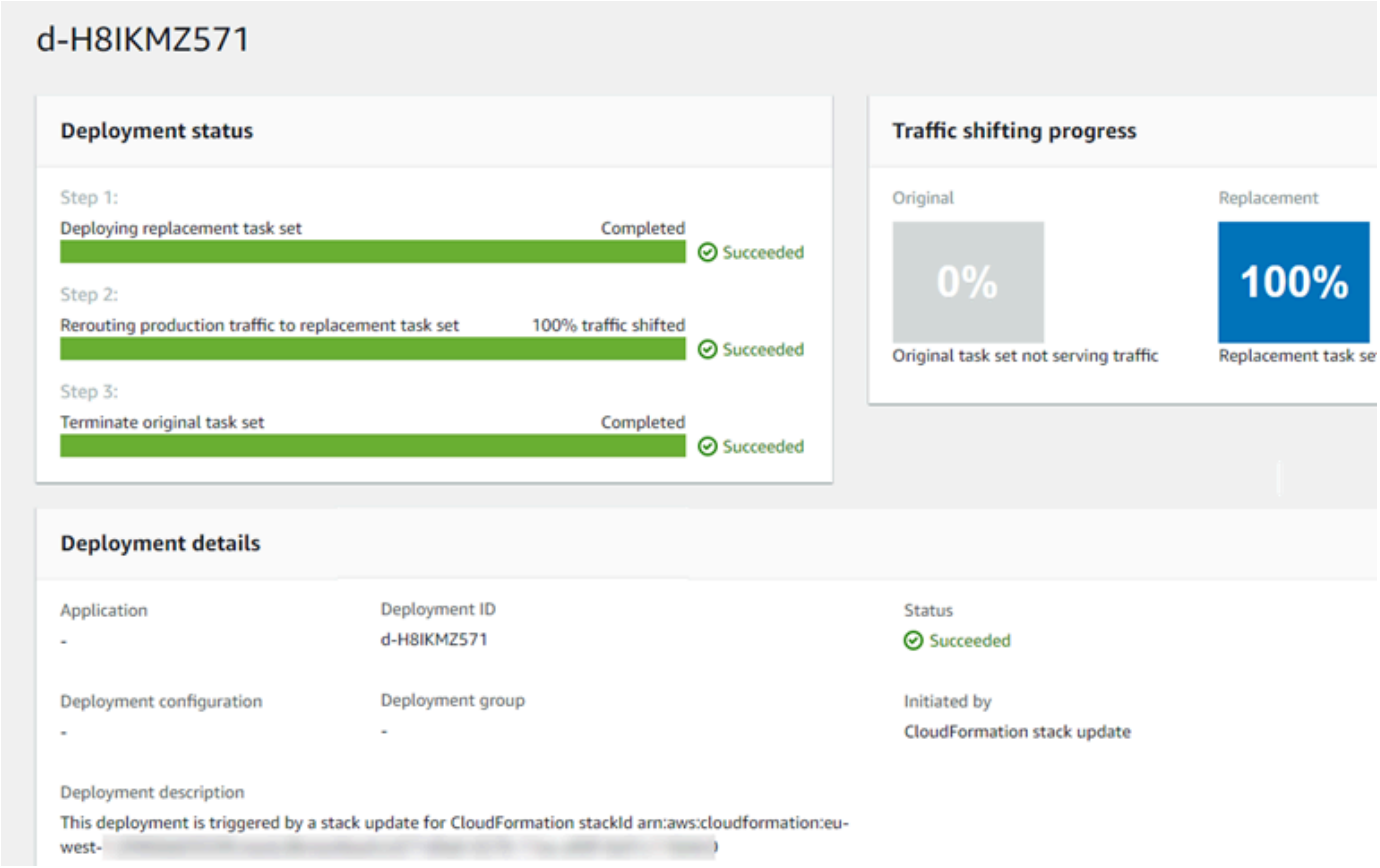
1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在部署中，會顯示由 AWS CloudFormation 堆疊更新觸發的部署。選擇部署以檢視 Deployment history (部署歷程記錄)。



3. 選擇部署以檢視流量轉移狀態。請注意，應用程式和部署群組並不會被建立。



4. 以下項目適用於復原或停用部署：

   - 成功的部署會顯示在中， CodeDeploy 並顯示部署是由初始化的 AWS CloudFormation。
   - 如果您要停止並復原部署，則必須在中取消堆疊更新 AWS CloudFormation。

檢視 CodeDeploy 部署詳情

您可以使用 CodeDeploy 主控台 AWS CLI、或 CodeDeploy API 來檢視與您 AWS 帳戶相關聯的部署詳細資料。

> **Note**
>
> 您可以在下列位置檢視執行個體上的 EC2 /內部部署記錄：
>
> - Amazon Linux，RHEL 和 Ubuntu 服務器：`/opt/codedeploy-agent/deployment-root/deployment-logs/codedeploy-agent-deployments.log`
> - 視窗伺服器:C:ProgramData\\ Amazon\ CodeDeploy <DEPLOYMENT-GROUP-ID><DEPLOYMENT-ID>\ 日誌\ scripts.log

如需詳細資訊，請參閱 分析日誌檔案以調查執行個體的部署失敗。

主題

- 檢視部署詳細資料 (主控台)
- 檢視部署詳細資料 (CLI)

檢視部署詳細資料 (主控台)

如果要使用主 CodeDeploy 控台檢視部署詳細資料：

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在瀏覽窗格中，展開 [部署]，然後選擇 [部署]。

   > **Note**
   >
   > 如果未顯示任何項目，請確定已選取正確的區域。在導覽列的區域選取器中，選擇「區域」和「端點」中列出的其中一個區域AWS 一般參考。 CodeDeploy 僅在這些地區支援。

3. 若要查看單一部署的其他詳細資訊，請在 Deployment history (部署歷程記錄) 中選擇部署 ID，或選擇部署 ID 旁的按鈕，然後選擇 View (檢視)。

檢視部署詳細資料 (CLI)

若要使用 AWS CLI 檢視部署詳細資料，請呼叫`get-deployment`命令或命`batch-get-deployments`令。您可以呼叫 `list-deployments` 命令取得唯一部署 ID 清單，做為 `get-deployment` 命令和 `batch-get-deployments` 命令的輸入。

若要檢視單一部署的詳細資料，請呼叫 get-deployment 命令，指定唯一的部署識別碼。若要取得部署識別碼，請呼叫清單部署命令。

若要檢視有關多個部署的詳細資料，請呼叫指batch-get-deployments令，指定多個唯一的部署識別碼。若要取得部署識別碼，請呼叫清單部署指令。

若要檢視部署識別碼的清單，請呼叫 list 部署指令，並指定：

- 與部署建立關聯的應用程式名稱。若要檢視應用程式名稱清單，請呼叫清單應用程式命令。
- 與部署建立關聯的部署群組名稱。若要檢視部署群組名稱清單，請呼叫命list-deployment-groups令。
- 選擇性地，是否包含依其部署狀態的部署詳細資訊 (如果未指定，將會列出所有相符的部署，不論其部署狀態為何)。
- 選擇性地，是否包含依其部署建立開始時間和 (或) 結束時間的部署詳細資訊 (如果未指定，將會列出所有相符的部署，不論其建立時間為何)。

檢視 CodeDeploy EC2/內部部署的記錄資料

您可以透過設定 Amazon CloudWatch 代理程式在 CloudWatch 主控台中檢視彙總資料，或登入個別執行個體以檢視日誌檔，以檢視 CodeDeploy 部署所建立的日誌資料。

> **Note**
>
> AWS Lambda 或 Amazon ECS 部署不支援日誌。它們只能針對 EC2 /內部部署建立。

主題

- 在 Amazon CloudWatch 主控台中檢視日誌檔資料
- 檢視執行個體上的記錄檔

在 Amazon CloudWatch 主控台中檢視日誌檔資料

將 Amazon CloudWatch 代理程式安裝在執行個體上時，該執行個體的所有部署資料都可以在 CloudWatch 主控台中檢視。為了簡單起見，我們建議您 CloudWatch 使用集中監視記錄檔，而不是逐個檢視記錄檔。如需詳細資訊，請參閱 傳送 CodeDeploy 代理程式記錄至 CloudWatch。

檢視執行個體上的記錄檔

若要檢視個別執行個體的部署日誌資料，您可以登入執行個體，並瀏覽錯誤或其他部署事件的相關資訊。

主題

若要檢視 Amazon Linux、RHEL 和 Ubuntu 伺服器執行個體上的部署日誌檔

在 Amazon Linux、RHEL 和 Ubuntu 伺服器執行個體上，部署日誌存放在下列位置：

`/opt/codedeploy-agent/deployment-root/deployment-logs/codedeploy-agent-deployments.log`

若要檢視或分析 Amazon Linux、RHEL 和 Ubuntu 伺服器執行個體上的部署日誌，請登入執行個體，然後輸入下列命令以開啟 CodeDeploy 代理程式記錄檔：

```
less /var/log/aws/codedeploy-agent/codedeploy-agent.log
```

輸入下列命令，以瀏覽日誌檔案中的錯誤訊息：

Command	結果
& ERROR	只在日誌檔案中顯示錯誤訊息。在 **ERROR** 文字前後使用單一空格。
/ ERROR	搜尋下一個錯誤訊息 [1]
? ERROR	搜尋先前的錯誤訊息。[2] 在字詞前後使用一個空格。**ERROR**
G	移至日誌檔案結尾。
g	移至日誌檔案開頭。
q	結束日誌檔案。

Command	結果
h	了解其他命令。

[1] 輸入後**/ ERROR** ，輸入下**n**一個錯誤訊息。輸入 **N** 表示前一個錯誤訊息。

[2] 輸入後**? ERROR** ，輸入下**n**一個錯誤訊息，或輸入上**N**一個錯誤訊息。

您也可以輸入下列命令來開啟指 CodeDeploy 令碼記錄檔：

```
less /opt/codedeploy-agent/deployment-root/deployment-group-ID/deployment-ID/logs/
scripts.log
```

輸入下列命令，以瀏覽日誌檔案中的錯誤訊息：

Command	結果
&stderr	只在日誌檔案中顯示錯誤訊息。
/stderr	搜尋下一個錯誤訊息 [1]
?stderr	搜尋先前的錯誤訊息 [2]
G	移至日誌檔案結尾。
g	移至日誌檔案開頭。
q	結束日誌檔案。
h	了解其他命令。

[1] 輸入後**/stderr**，輸入下**n**一則錯誤訊息。輸入 **N** 表示將前一個錯誤訊息往回。

[2] 輸入後**?stderr**，輸入**n**下一個錯誤訊息。輸入 **N** 表示將前一個錯誤訊息往前。

若要檢視 Windows 伺服器執行個體上的部署記錄檔

CodeDeploy 代理程式記錄檔：在 Windows Server 執行個體上， CodeDeploy 代理程式記錄檔儲存在下列位置：

`C:\ProgramData\Amazon\CodeDeploy\log\codedeploy-agent-log.txt`

若要檢視或分析 Windows Server 執行個體上的 CodeDeploy 代理程式記錄檔，請登入執行個體，然後輸入下列命令以開啟檔案：

```
notepad C:\ProgramData\Amazon\CodeDeploy\log\codedeploy-agent-log.txt
```

若要瀏覽日誌檔案中的錯誤訊息，請按 CTRL+F，並輸入 **ERROR [**，然後按 Enter 找到第一個錯誤。

CodeDeploy 指令碼記錄檔：在 Windows Server 執行個體上，部署記錄會儲存在下列位置：

`C:\ProgramData\Amazon\CodeDeploy\`*`deployment-group-id`*`\`*`deployment-id`*`\logs\scripts.log`

其中：

- *`deployment-group-id`*是一個字符串，如 `examplebf3a9c7a-7c19-4657-8684-b0c68d0cd3c4`
- *`deployment-id`* 是 `d-12EXAMPLE` 這類識別符

輸入下列命令以開啟指 CodeDeploy 令碼記錄檔：

```
notepad C:\ProgramData\Amazon\CodeDeploy\deployment-group-ID\deployment-ID\logs
\scripts.log
```

若要瀏覽日誌檔案中的錯誤訊息，請按 CTRL+F，並輸入 **stderr**，然後按 Enter 找到第一個錯誤。

停止部署 CodeDeploy

您可以使用 CodeDeploy 主控台 AWS CLI、或 CodeDeploy API 來停止與您 AWS 帳戶相關聯的部署。

> **Warning**
>
> 停止 EC2 /內部部署可使部署群組的部分或所有執行個體處於不確定的部署狀態。如需詳細資訊，請參閱 已停止和失敗的部署。

您可以停止部署，也可以停止並復原部署。

- 停止部署 (主控台)
- 停止部署 (CLI)

> **Note**
>
> 如果您的部署是透過藍/綠部署 AWS CloudFormation，則無法在 CodeDeploy 主控台中執行此工作。轉到 AWS CloudFormation 控制台以執行此任務。

停止部署 (主控台)

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在瀏覽窗格中，展開 [部署]，然後選擇 [部署]。

   > **Note**
   >
   > 如果未顯示任何項目，請確定已選取正確的區域。在導覽列的區域選取器中，選擇「區域」和「端點」中列出的其中一個區域AWS 一般參考。 CodeDeploy 僅在這些地區支援。

3. 選擇您想要停止的部署，執行下列其中一項：

   1. 選擇 Stop deployment (停止部署)，以停止部署而不轉返。
   2. 選擇 Stop and roll back deployment (停止並轉返部署)，以停止並轉返部署。

   如需詳細資訊，請參閱 使用以下方式重新部署和復原部署 CodeDeploy。

> **Note**
>
> 如果 Stop deployment (停止部署) 和 Stop and roll back deployment (停止並轉返部署) 無法使用，表示部署已經進行到無法停止的點。

停止部署 (CLI)

呼叫停止部署命令，指定部署識別碼。若要檢視部署識別碼的清單，請呼叫清單部署指令。

使用以下方式重新部署和復原部署 CodeDeploy

CodeDeploy 將先前部署的應用程式修訂版重新部署為新部署，以復原部署。這些轉返的部署在技術上而言是具有新部署 ID 的新部署，而不是先前部署的還原版本。

您可以自動或手動轉返部署。

主題

- 自動回復
- 手動復原
- 復原和重新部署工作流程
- 復原現有內容的行為

自動回復

您可以設定部署群組，或設定部署在部署失敗或到達您指定的監控閾值時自動轉返。在此案例下，便會部署最後一個已知良好的應用程式修訂版本。當您建立應用程式或是建立或更新部署群組時，可以設定自動轉返。

當您建立新的部署時，您也可以選擇覆寫先前為部署群組指定的自動轉返組態。

> **Note**
>
> 您可以使用 Amazon 簡單通知服務，在自動復原部署時接收通知。如需相關資訊，請參閱Monitoring Deployments with Amazon SNS Event Notifications。

如需設定自動轉返的詳細資訊，請參閱設定部署群組的進階選項。

手動復原

如果您尚未設定自動轉返，則可以建立使用任何先前部署應用程式修訂的新應用程式，並遵循重新部署修訂的步驟，以手動轉返部署。如果應用程式已進入不明狀態，則您可能會這麼做。您可以將應用程式重新部署已知工作中狀態，而不是花太多時間來故障診斷。如需詳細資訊，請參閱 使用建立部署 CodeDeploy。

> **Note**
>
> 如果您從部署群組中移除執行個體，則 CodeDeploy 不會解除安裝任何可能已安裝在該執行個體上的執行個體。

復原和重新部署工作流程

啟動自動復原時，或手動啟動重新部署或手動復原時，會 CodeDeploy 先嘗試從每個參與的執行處理移除上次成功安裝的所有檔案。 CodeDeploy 通過檢查清理文件來做到這一點：

`/opt/codedeploy-agent/deployment-root/deployment-instructions/`*`deployment-group-ID`*`-cleanup`檔案 (適用於 Amazon Linux、Ubuntu 伺服器和 RHEL 執行個體)

`C:\ProgramData\Amazon\CodeDeploy\deployment-instructions\`*`deployment-group-ID`*`-cleanup`檔案 (適用於 Windows 伺服器執行個體)

如果存在，則在開始新部署之前， CodeDeploy 使用清理檔案從例證中移除所有列示的檔案。

例如，前兩個文字檔案和兩個指令碼檔案已部署到執行 Windows Server 的 Amazon EC2 執行個體，而指令碼在部署生命週期事件期間建立了兩個文字檔案：

```
c:\temp\a.txt (previously deployed by CodeDeploy)
c:\temp\b.txt (previously deployed by CodeDeploy)
c:\temp\c.bat (previously deployed by CodeDeploy)
c:\temp\d.bat (previously deployed by CodeDeploy)
c:\temp\e.txt (previously created by c.bat)
c:\temp\f.txt (previously created by d.bat)
```

清理檔案只會列出前兩個文字檔案和兩個指令碼檔案：

```
c:\temp\a.txt
```

```
c:\temp\b.txt
c:\temp\c.bat
c:\temp\d.bat
```

在新部署之前，只 CodeDeploy 會移除前兩個文字檔和兩個指令碼檔案，保留最後兩個文字檔案不變：

```
c:\temp\a.txt will be removed
c:\temp\b.txt will be removed
c:\temp\c.bat will be removed
c:\temp\d.bat will be removed
c:\temp\e.txt will remain
c:\temp\f.txt will remain
```

在此程序中，不 CodeDeploy 會嘗試還原或以其他方式協調先前部署中任何指令碼在後續重新部署期間所採取的任何動作，無論是手動還是自動復原。例如，如果`c.bat`和`d.bat`檔案包含不重新建立`e.txt`和檔`f.txt`案的邏輯 (如果它們已存在)，則舊版本的`e.txt`和`f.txt`將在每次 CodeDeploy 執行`c.bat`和後續部署`d.bat`中保持不變。您可以將邏輯新增至 `c.bat` 和 `d.bat` 一律檢查 `e.txt` 和 `f.txt` 的舊版本，並在建立新的版本之前予以刪除。

復原現有內容的行為

作為部署程序的一部分， CodeDeploy 代理程式會從每個執行個體中移除最近部署所安裝的所有檔案。如果不屬於先前部署的檔案出現在目標部署位置中，您可以選擇在下次部署期間如 CodeDeploy 何處理這些檔案：

- 部署失敗 — 報告錯誤，且部署狀態變更為「失敗」。
- 覆寫內容 — 應用程式修訂版本中的檔案版本會取代實例上已有的版本。
- 保留內容 — 保留目標位置中的檔案，且不會將應用程式修訂版本中的版本複製到實例。

您可以在建立部署時選擇此行為。如果在主控台中建立部署，請參閱建立 EC2/內部部署計算平台部署 (主控台)。如果使用建立部署 AWS CLI，請參閱建立EC2/內部部署計算平台部署 (CLI)。

您可以選擇保留您下一個部署想要含有的檔案，而不需要將這些檔案新增至應用程式修訂套件。例如，您將檔案直接上傳至部署所需的執行個體，但未新增至應用程式修訂套件。或者，如果您的應用程式已經在生產環境中，但您想要第一次使用 CodeDeploy 來部署檔案，則可以將檔案上傳到執行個體。

如果發生因部署失敗而重新部署最新成功部署應用程式修訂的轉返，則會將該最後成功部署的內容處理選項套用至轉返部署。

不過，如果失敗的部署設定為覆寫，而不是保留檔案，則在轉返期間可能發生意外結果。具體而言，失敗的部署可能會移除您預期保留的檔案。轉返部署執行時，檔案不在執行個體上。

在下列範例中，有三種部署。在部署 3 期間再次部署應用程式修訂 1 時，任何在失敗的第二個部署期間覆寫 (刪除) 的檔案都無法再使用 (無法保留)：

部署	應用程式修訂	內容覆寫選項	部署狀態	行為和結果
部署 1	應用程式修訂 1	RETAIN (保留)	Succeeded	CodeDeploy 偵測目標位置中未由先前部署部署所部署的檔案。這些檔案可能故意放在該處，而成為目前部署的一部分。它們會保留並記錄為目前部署套件的一部分。
部署 2	應用程式修訂 2	OVERWRITE (覆寫)	失敗	在部署程序期間，CodeDeploy 會刪除上一次成功部署的所有檔案。這包含在部署 1 期間保留的檔案。 不過，部署因無關的原因而失敗。
部署 3	應用程式修訂 1	RETAIN (保留)		由於部署或部署群組已啟用 auto 復原，CodeDeploy因此請部署最後一個已知的良好應用

部署	應用程式修訂	內容覆寫選項	部署狀態	行為和結果
				程式修訂版本，即應用程式修訂版本 1。 但是，在部署 2 失敗之前，您要保留在部署 1 中的檔案已刪除，且無法由其擷取 AWS CodeDeploy。您可以自行將它們新增至執行個體 (如果應用程式修訂 1 需要它們)，也可以建立新的應用程式修訂。

在不同的 AWS 帳戶中部署應用程式

Organizations 通常會有多個用於不同用途的 AWS 帳戶 (例如，一個用於系統管理工作，另一個用於開發、測試和生產工作，或一個與開發和測試環境相關聯，另一個與生產環境相關聯)。

雖然您可能會在不同的帳戶中執行相關工作，但 CodeDeploy 部署群組和其部署的 Amazon EC2 執行個體會嚴格關聯到建立這些執行個體的帳戶。無法作的事如，新增在帳戶啟動的執行個體至其他帳戶的部署群組中。

假設您有兩個 AWS 帳戶：您的開發帳戶和生產帳戶。您主要在開發帳戶中工作，但您希望能夠在生產帳戶中啟動部署，同時不需要完整的登入資料確認，也不用登出開發帳戶和登入生產帳戶。

遵循跨帳戶組態步驟後，您可以啟動屬於您組織的另一個帳戶的部署，同時無需使用該帳戶的完整登入資料。您可以執行此操作，部分是使用 AWS Security Token Service (AWS STS) 提供的功能，該功能授予您對該帳戶的臨時訪問權限。

步驟 1：在任一帳戶中建立 S3 儲存貯體

無論是開發帳戶或生產帳戶：

- 如果您尚未這麼做，請建立 Amazon S3 儲存貯體，在其中存放生產帳戶的應用程式修訂版本。如需詳細資訊，請參閱在 Amazon S3 中建立儲存貯體。您甚至可以讓兩個帳戶使用相同的儲存貯體以及應用程式修改版，部署相同檔案到您的生產環境，讓您測試和驗證開發帳戶。

步驟 2：將 Amazon S3 儲存貯體許可授與生產帳戶的 IAM 執行個體設定檔

如果您在步驟 1 中建立的 Amazon S3 儲存貯體位於您的生產帳戶中，則不需要執行此步驟。稍後您假設的角色可以存取此儲存貯體，因為它也在生產帳戶中了。

如果您在開發帳戶中建立 Amazon S3 儲存貯體，請執行下列動作：

- 在生產帳戶中，建立 IAM 執行個體設定檔。如需相關資訊，請參閱步驟 4：為您的 Amazon EC2 執行個體建立 IAM 執行個體設定檔。

  > **Note**
  >
  > 記下此 IAM 執行個體設定檔的 ARN。您將需要新增它到您接下來建立的跨儲存貯體政策中。

- 在開發帳戶中，將您在開發帳戶中建立的 Amazon S3 儲存貯體存取權授與您剛在生產帳戶中建立的 IAM 執行個體設定檔。如需詳細資訊，請參閱範例 2：授與跨帳戶值區權限的值區擁有者。

  在完成授予跨帳戶儲存貯體權限許可的過程中，請注意以下事項：

  - 在範例攻略中，帳戶 A 代表您的開發帳戶，帳戶 B 代表您的生產帳戶。
  - 當您 執行帳戶 A (開發帳戶) 任務、修改以下儲存貯體政策以授與跨帳戶許可，而不使用範例政策，請參閱逐步解說。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Cross-account permissions",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::account-id:role/role-name"
```

```
            },
            "Action": [
               "s3:Get*",
               "s3:List*"
            ],
            "Resource": [
               "arn:aws:s3:::bucket-name/*"
            ]
         }
      ]
}
```

account t-id 代表您剛建立 IAM 執行個體設定檔的生產帳戶的帳戶編號。

*####*代表您剛建立的 IAM 執行個體設定檔的名稱。

bucket-name 表示您在步驟一中建立的儲存貯體名稱。請確認在您的儲存貯體名稱後含有的 /*，提供對儲存貯體內每個文件的存取權限。

步驟 3：在生產帳戶中建立資源與跨帳戶角色

在您的生產帳戶中：

- 使用本指南中的指示建立 CodeDeploy 資源，包括應用程式、部署群組、部署組態、Amazon EC2 執行個體、Amazon EC2 執行個體設定檔、服務角色等。
- 建立其他角色 (跨帳戶 IAM 角色)，讓開發帳戶中的使用者可以假設在此生產帳戶中執行 CodeDeploy 作業。

  使用逐步解說：使用 IAM 角色指導跨 AWS 帳戶委派存取權，以協助您建立跨帳戶角色。除了將逐步解說中的範例權限新增至您的政策文件，您應該至少將下列兩個 AWS 提供的原則附加至該角色：

  - `AmazonS3FullAccess`：僅在 S3 儲存貯體位於開發帳戶時才需要。提供假定的生產帳戶角色，以完整存取開發帳戶中 Amazon S3 服務和資源 (其中存放修訂)。
  - `AWSCodeDeployDeployerAccess`：可讓使用者註冊和部署修訂。

  如果您想要建立以及管理部署群組以及不只是初始化部署，新增 `AWSCodeDeployFullAccess` 原則而不是 `AWSCodeDeployDeployerAccess` 原則。如需使用 IAM 受管政策授與 CodeDeploy 任務許可的詳細資訊，請參閱AWS 的管理 (預先定義) 策略 CodeDeploy。

  您可以附加原則，如果您想要在其他 AWS 服務執行工作，使用此跨帳戶角色。

> **Important**
>
> 建立跨帳戶 IAM 角色時，請記下存取生產帳戶所需的詳細資料。
> 若要使用 AWS Management Console 來切換角色，您必須提供下列其中一項：
>
> - URL 用於啟動生產帳戶與假設角色認證。您將在 檢視 頁面上找到URL，其顯示在跨帳戶角色建立程序的末端。
> - 跨帳戶角色的名稱及帳戶 ID 數或別名都可以。
>
> 若要使用 AWS CLI 來切換角色，您必須提供下列項目：
>
> - 跨帳戶角色的 ARN，您將假設。

步驟 4：將應用程式修訂版上傳到 Amazon S3 儲存貯體

在您建立 Amazon S3 儲存貯體的帳戶中：

- 將您的應用程式修訂版上傳到 Amazon S3 儲存貯體。如需相關資訊，請參閱將修訂推送 CodeDeploy 至 Amazon S3 (僅適用於 EC2 /內部部署)。

步驟 5：假設跨帳戶角色並部署應用程式

在開發帳戶中，您可以使用 AWS CLI 或 AWS Management Console 來承擔跨帳戶角色，並在生產帳戶中啟動部署。

如需有關如何使用切 AWS Management Console 換角色和初始化部署的指示，請參閱切換至角色 (AWS Management Console) 和建立 EC2/內部部署計算平台部署 (主控台)。

如需如何使用 AWS CLI 來擔任跨帳戶角色和啟動部署的指示，請參閱切換至 IAM 角色 (AWS Command Line Interface) 和建立EC2/內部部署計算平台部署 (CLI)。

若要取得有關使用角色的更多資訊 AWS STS，請參閱《AWS Security Token Service 使用指南》AssumeRole中的〈指AWS CLI 令參考〉中的〈assume-role〉中的〈〉。

相關主題：

- CodeDeploy：從開發帳戶部署到生產帳戶

使用 CodeDeploy 代理程式驗證本機電腦上的部署套件

您可以使用 CodeDeploy 代理程式在登入的執行個體上部署內容。這可讓您測試要在部署中使用的應用程式規格AppSpec 檔案 (檔案) 的完整性，以及要部署的內容。

您不需要建立應用程式以及部署群組。如果您想要部署儲存在本機執行個體上的內容，您甚至不需要 AWS 帳戶。對於最簡單的測試，您可以在包含 AppSpec 文件和要部署的內容的目錄中運行codedeploy-local命令，而無需指定任何選項。工具中有其他測試案例的選項。

您可以在本機上驗證部屬包裹。

- 測試應用程式修訂版的完整性。
- 測試 AppSpec 檔案的內容。
- 第一 CodeDeploy 次嘗試使用現有的應用程式程式碼。
- 當您已經登入執行個體，快速部署內容。

您可以使用存放在本機執行個體或支援的遠端存放庫類型 (Amazon S3 儲存貯體或公有儲存 GitHub 庫) 中的部署內容。

必要條件

您開始就地部署之前，請完成下列步驟：

- 建立或使用 CodeDeploy 代理程式支援的執行個體類型。如需相關資訊，請參閱 CodeDeploy 代理程式支援的作業系統。
- 安裝代理程式版本 1.0.1.1352 或更新版本。 CodeDeploy 如需相關資訊，請參閱安裝 CodeDeploy 代理程式。
- 如果您要從 Amazon S3 儲存貯體或 GitHub 儲存庫部署內容，請佈建要與之搭配使用的使用者 CodeDeploy。如需相關資訊，請參閱步驟 1：設定。
- 如果您要從 Amazon S3 儲存貯體部署應用程式修訂版，請在您工作的區域建立 Amazon S3 儲存貯體，並將 Amazon S3 儲存貯體政策套用至儲存貯體。此政策會將下載應用程式修訂所需的許可授予您的執行個體。

  例如，下列 Amazon S3 儲存貯體政策允許任何具有附加 IAM 執行個體設定檔 (包含 ARN`arn:aws:iam::444455556666:role/CodeDeployDemo`) 的 Amazon EC2 執行個體，從 Amazon S3 儲存貯體的任何位置下載名為`codedeploydemobucket`：

```
{
```

```
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::codedeploydemobucket/*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666:role/CodeDeployDemo"
                ]
            }
        }
    ]
}
```

下列 Amazon S3 儲存貯體政策允許任何具codedeploydemobucket有相關聯 IAM 使用者 (包含 ARNarn:aws:iam::444455556666:user/CodeDeployUser) 的現場部署執行個體從 Amazon S3 儲存貯體中的任何位置下載：

```
{
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::codedeploydemobucket/*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666:user/CodeDeployUser"
                ]
            }
        }
    ]
}
```

如需如何產生和連接 Amazon S3 儲存貯體政策的詳細資訊，請參閱儲存貯體政策範例。

- 如果您要從 Amazon S3 儲存貯體或 GitHub儲存庫部署應用程式修訂版，請設定 IAM 執行個體設定檔並將其附加到執行個體。如需詳細資訊，請參閱 步驟 4：為您的 Amazon EC2 執行個體建立 IAM 執行個體設定檔、為 CodeDeploy （AWS CLI 或 Amazon EC2 控制台）創建一個 Amazon EC2 實例 及 為 CodeDeploy (AWS CloudFormation 範本) 建立 Amazon EC2 執行個體。
- 如果您要從中部署內容 GitHub，請建立 GitHub 帳戶和公用存放庫。若要建立 GitHub 帳戶，請參閱加入 GitHub。若要建立 GitHub儲存庫，請參閱建立存放庫。

> **Note**
>
> 目前不支援私人儲存庫。如果您的內容儲存在私人存 GitHub 放庫中，您可以將其下載至執行個體，並使用`--bundle-location`選項來指定其本機路徑。

- 準備要部署到執行個體的內容 (包括 AppSpec 檔案)，並將其放置在本機執行個體、Amazon S3 儲存貯體或 GitHub 儲存庫中。如需相關資訊，請參閱使用的應用程式修訂 CodeDeploy。
- 如果您想要使用其他組態選項的預設值以外的值，請建立組態檔案並將其放置在執行個體上 (`/etc/codedeploy-agent/conf/codedeployagent.yml`適用於 Amazon Linux、RHEL 或 Ubuntu 伺服器執行個體，或適用`C:\ProgramData\Amazon\CodeDeploy\conf.yml`於 Windows 伺服器執行個體)。如需相關資訊，請參閱CodeDeploy 用戶端組態參考。

> **Note**
>
> 如果您在 Amazon Linux、RHEL 或 Ubuntu 伺服器執行個體上使用組態檔案，您必須：
>
> - 使用 `:root_dir:` 和 `:log_dir:` 變數指定部署根以及紀錄目錄資料夾預設以外的地點。
> - 用`sudo`於執行 CodeDeploy 代理程式命令。

建立本機部署

在您要建立本機部署的執行個體上，開啟終端機工作階段 (Amazon Linux、RHEL 或 Ubuntu 伺服器執行個體) 或命令提示字元 (Windows 伺服器) 以執行工具命令。

> **Note**
>
> codedeploy-local 命令安裝於以下位置：
>
> - 在 Amazon Linux 上，RHEL 或 Ubuntu 服務器：`/opt/codedeploy-agent/bin`。

- 在視窗伺服器上：C:\ProgramData\Amazon\CodeDeploy\bin.

基本命令語法

```
codedeploy-local [options]
```

概要

```
codedeploy-local
[--bundle-location <value>]
[--type <value>]
[--file-exists-behavior <value>]
[--deployment-group <value>]
[--events <comma-separated values>]
[--agent-configuration-file <value>]
[--appspec-filename <value>]
```

選項

-l ， - 配套位置

應用程式修訂版配套位置。如果您不指定一個位置，則該工具以您目前作業的預設使用目錄。如果您指定一個值給 --bundle-location，則您必須也指定一個值給 --type。

配套位置格式化範例：

- 本地 Amazon Linux、RHEL 或 Ubuntu 伺服器執行個體：/path/to/local/bundle.tgz
- 本地視窗伺服器實例：C:/path/to/local/bundle
- Amazon S3 桶：s3://mybucket/bundle.tar
- GitHub 儲存庫：https://github.com/*account-name*/*repository-name*/

-t 、 - 類型

應用程式修訂版配套格式。支援的類型包括 tgz、tar、zip 和directory。如果您未指定一個形式，工具會使用 directory 作為預設。如果您指定一個值給 --type，則您必須也指定一個值給 --bundle-location。

-b, --file-exists-behavior

指出檔案如何處理已經存在部署標的位置裡，但不是成功部署上一步的一部分。選項包含不允許、覆寫、保留。如需詳細資訊，請參閱 AWS CodeDeploy API 參考fileExistsBehavior中的。

-g、- 部署群組

目標位置的資料夾路徑為待部署的內容。如果您未指定資料夾，工具會在您的部署根目錄default-local-deployment-group中建立一個名為的資料夾。針對每一個您建立的區域部署，該工具會在子目錄裡建立名稱如 d-98761234-local 的資料夾。

-e、 - 事件

您要依序執行的一組取代生命週期事件掛接，而不是您在檔案中列示的事 AppSpec 件。多個關聯可以被指定，以逗號分隔。您可以使用這個選項，如果：

- 您想要執行一組不同的事件，而不需要更新 AppSpec 檔案。
- 您想要執行單一事件勾點，做為 AppSpec 檔案中內容的例外狀況，例如`ApplicationStop`。

如果您未在覆寫清單中指定DownloadBundle和 Install 事件，它們會在您指定的所有事件掛接之前執行。如果您在`--events`選項清單中包含DownloadBundle並安裝，則它們必須只有通常在 CodeDeploy 部署中執行之前執行的事件。如需相關資訊，請參閱AppSpec 「掛鉤」部分。

-c, --agent-configuration-file

如果您將它存放至預設位置以外的地方，本機將使用於部署的設定檔案。設定檔案指定替代到其他預設值以及部署表現方式。

依預設，組態檔案會儲存在 `/etc/codedeploy-agent/conf/codedeployagent.yml` (Amazon Linux、RHEL 或 Ubuntu 伺服器執行個體) 或 `C:/ProgramData/Amazon/CodeDeploy/conf.yml` (視窗伺服器)。如需詳細資訊，請參閱 CodeDeploy 用戶端組態參考。

-A，應用程序規格文件名

檔案的 AppSpec 名稱。對於本機部署，接受的值為`appspec.yml`和`appspec.yaml`。依預設，會呼叫 AppSpec 檔案`appspec.yml`。

-h、 -求助

顯示協助內容的摘要。

-v、 - 版本

顯示工具版本編號。

範例

以下是有效命令格式的範例。

```
codedeploy-local
```

```
codedeploy-local --bundle-location /path/to/local/bundle/directory
```

```
codedeploy-local --bundle-location C:/path/to/local/bundle.zip --type zip --deployment-group my-deployment-group
```

```
codedeploy-local --bundle-location /path/to/local/directory --type directory --deployment-group my-deployment-group
```

從 Amazon S3 部署服務包：

```
codedeploy-local --bundle-location s3://mybucket/bundle.tgz --type tgz
```

```
codedeploy-local --bundle-location s3://mybucket/bundle.zip?versionId=1234&etag=47e8 --type zip --deployment-group my-deployment-group
```

從公用 GitHub 存放庫部署套裝軟體：

```
codedeploy-local --bundle-location https://github.com/awslabs/aws-codedeploy-sample-tomcat --type zip
```

```
codedeploy-local --bundle-location https://api.github.com/repos/awslabs/aws-codedeploy-sample-tomcat/zipball/master --type zip
```

```
codedeploy-local --bundle-location https://api.github.com/repos/awslabs/aws-codedeploy-sample-tomcat/zipball/HEAD --type zip
```

```
codedeploy-local --bundle-location https://api.github.com/repos/awslabs/aws-codedeploy-sample-tomcat/zipball/1a2b3c4d --type zip
```

部署一個配套指定多個生命週期事件：

```
codedeploy-local --bundle-location /path/to/local/bundle.tar --type tar --application-
folder my-deployment --events DownloadBundle,Install,ApplicationStart,HealthCheck
```

使用 ApplicationStop 生命週期事件停止先前部署的應用程式：

```
codedeploy-local --bundle-location /path/to/local/bundle.tgz --type tgz --deployment-
group --events ApplicationStop
```

一個指定部署群組 ID 進行部署：

```
codedeploy-local --bundle-location C:/path/to/local/bundle/directory --deployment-group
 1234abcd-5dd1-4774-89c6-30b107ac5dca
```

```
codedeploy-local --bundle-location C:/path/to/local/bundle.zip --type zip --deployment-
group 1234abcd-5dd1-4774-89c6-30b107ac5dca
```

監控部署 CodeDeploy

監控是維持 AWS 解決方案的可靠性、可用性和效能的 CodeDeploy 重要組成部分。您應該從 AWS 解決方案的所有部分收集監視資料，以便在發生多點失敗時更輕鬆地偵錯。但是 CodeDeploy，在開始監視之前，您應該先建立監視計劃，其中包含下列問題的答案：

- 監控目標是什麼？
- 要監控哪些資源？
- 監控這些資源的頻率為何？
- 要使用哪些監控工具？
- 誰將執行監控任務？
- 發生問題時應該通知誰？

下一個步驟是透過測量不同時間和不同負載條件下的效能，建立環境中正常 CodeDeploy 效能的基準。監視時 CodeDeploy，請儲存歷史監視資料，以便您可以將其與目前的效能資料進行比較，識別正常的效能模式和效能異常，並設計解決問題的方法。

例如，如果您正在使用 CodeDeploy，則可以監視部署和目標執行個體的狀態。當部署或執行個體失敗時，您可能需要重新設定應用程式規格檔案、重新安裝或更新 CodeDeploy 代理程式、更新應用程式或部署群組中的設定，或變更執行個體設定或 AppSpec 檔案。

若要建立基準，您至少必須監控下列項目：

- 部署事件和狀態
- 執行個體事件和狀態

自動化監控工具

AWS 提供了可用於監視的各種工具 CodeDeploy。您可以設定其中一些工具來進行監控，但有些工具需要手動介入。建議您盡可能自動化監控任務。

您可以使用以下自動監視工具來觀看 CodeDeploy 和報告出現問題時：

- Amazon CloudWatch 警示 — 觀看您指定期間內的單一指標，並根據指定臨界值在多個時段內相對於指定閾值的指標值執行一或多個動作。動作是傳送至亞馬遜簡單通知服務 (Amazon SNS) 主題

或 Amazon EC2 Auto Scaling 政策的通知。 CloudWatch 警示不會僅因為處於特定狀態而叫用動作；狀態必須已變更並維持指定數目的期間。如需詳細資訊，請參閱 Monitoring Deployments with Amazon CloudWatch Tools。

如需更新服務角色以使用 CloudWatch 警示監視的相關資訊，請參閱 CloudWatch 授與 CodeDeploy 服務角色的權限。如需將 CloudWatch 警示監控新增至 CodeDeploy 作業的資訊建立應用程式 CodeDeploy，請參閱建立部署群組 CodeDeploy、或變更部署群組設定 CodeDeploy。

- Amazon CloudWatch 日誌 — 監控、存放和存取來自 AWS CloudTrail 或其他來源的日誌檔。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的監控日誌檔。

  如需使用 CloudWatch 主控台檢視 CodeDeploy記錄檔的詳細資訊，請參閱在 CodeDeploy 記錄主控台中檢視 CloudWatch 記錄檔。
- Amazon E CloudWatch vents — 匹配事件並將其路由到一個或多個目標函數或串流，以進行變更、擷取狀態資訊並採取糾正措施。有關更多信息，請參閱 Amazon 用 CloudWatch 戶指南中的 Amazon CloudWatch 事件是什麼。

  如需有關在作業中使用 CloudWatch 事件的 CodeDeploy 資訊，請參閱使用 Amazon CloudWatch 事件監控部署。
- AWS CloudTrail 記錄監控 — 在帳戶之間共用記錄檔、即時監控記 CloudTrail 錄檔案，方法是將記錄檔傳送至CloudWatch 記錄檔、使用 Java 撰寫記錄處理應用程式，以及驗證您的記錄檔在傳送之後未變更 CloudTrail。若要取得更多資訊，請參閱《使用指南》中的〈AWS CloudTrail 使用 CloudTrail 記錄檔〉。

  如需 CloudTrail 搭配使用的資訊 CodeDeploy，請參閱Monitoring Deployments。
- Amazon 簡單通知服務 — 設定事件驅動的觸發器，以接收有關部署和執行個體事件 (例如成功或失敗) 的 SMS 或電子郵件通知。如需詳細資訊，請參閱建立主題和什麼是 Amazon 簡單通知服務。

  如需設定的 Amazon SNS 通知的相關資訊 CodeDeploy，請參閱Monitoring Deployments with Amazon SNS Event Notifications。

手動監控工具

監視 CodeDeploy 的另一個重要部分是手動監視 CloudWatch 警報未涵蓋的項目。 CodeDeploy CloudWatch、和其他 AWS 主控台儀表板可提供您 AWS 環境狀態的 at-a-glance 檢視。我們建議您也檢查部 CodeDeploy 署時的記錄檔。

- CodeDeploy 控制台顯示：

  - 部署的狀態
  - 最近一次嘗試和最後一次成功部署修訂版的日期和時間。
  - 執行個體成功、失敗、略過，或部署中的數量。
  - 現場部署執行個體的狀態
  - 現場部署執行個體註冊或撤銷註冊的日期和時間。
- CloudWatch 主頁顯示：
  - 目前警示與狀態
  - 警示與資源的圖表
  - 服務運作狀態

  此外，您可以使用執行 CloudWatch 以下操作：
  - 建立自定儀表板來監控您注重的服務
  - 用於疑難排解問題以及探索驅勢的圖形指標資料。
  - 搜尋並瀏覽所有資 AWS 源指標
  - 建立與編輯要通知發生問題的警示

主題

- Monitoring Deployments with Amazon CloudWatch Tools
- Monitoring Deployments
- Monitoring Deployments with Amazon SNS Event Notifications

使用 Amazon CloudWatch 工具監控部署

您可以使用下列 CloudWatch 工具監控 CodeDeploy 部署：Amazon CloudWatch 事件、 CloudWatch 警示和 Amazon CloudWatch 日誌。

檢閱 CodeDeploy 代理程式和部署建立的記錄檔可協助您疑難排解部署失敗的原因。除了一次檢視一個執行個體的 CodeDeploy 記錄檔外，您還可以使用 CloudWatch Logs 監控集中位置的所有記錄。

如需使用 CloudWatch 警示和 CloudWatch 事件監視 CodeDeploy 部署的相關資訊，請參閱下列主題。

主題

- 使用 CloudWatch 警示監控部署 CodeDeploy

- 使用 Amazon CloudWatch 事件監控部署

使用 CloudWatch 警示監控部署 CodeDeploy

您可以為您在 CodeDeploy操作中使用的執行個體或 Amazon EC2 Auto Scaling 群組建立 CloudWatch 警示。警示會監看所指定時段內的單一指標，並根據與多個時段內給定閾值相對的指標值來執行一或多個動作。 CloudWatch 警報會在狀態變更時叫用動作 (例如，從OK到ALARM)。

使用原生 CloudWatch 警示功能，您可以指定在部署中使用的執行個體失敗 CloudWatch 時支援的任何動作，例如傳送 Amazon SNS 通知或停止、終止、重新啟動或復原執行個體。對於您的 CodeDeploy 作業，您可以將部署群組設定為在啟動與部署群組關聯的任何 CloudWatch 警示時停止部署。

您最多可以將十個 CloudWatch 警示與 CodeDeploy 部署群組產生關聯。如果任何指定的警示已啟動、部署停止、且狀態已更新為停用。若要使用此選項，您必須授與 CodeDeploy 服務角色的 CloudWatch 權限。

如需在 CloudWatch 主控台中設定 CloudWatch 警示的相關資訊，請參閱 Amazon CloudWatch 使用者指南中的建立 Amazon CloudWatch 警示。

如需關聯 CloudWatch 警示與中的部署群組的資訊 CodeDeploy，請參閱建立部署群組 CodeDeploy和變更部署群組設定 CodeDeploy。

主題

- CloudWatch 授與 CodeDeploy 服務角色的權限

CloudWatch 授與 CodeDeploy 服務角色的權限

在您的部署中使用 CloudWatch 警示監控之前，必須先授與您在 CodeDeploy 作業中使用的服務角色存取 CloudWatch 資源的權限。

授與服務角色的 CloudWatch 權限

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在 IAM 主控台的導覽窗格中，選擇 [角色]。
3. 選擇您在 AWS CodeDeploy 作業中使用的服務角色名稱。
4. 在 Permissions (許可) 標籤的 Inline Policies (內嵌政策) 區域中，選擇 Create Role Policy (建立角色政策)。

— 或 —

若 Create Role Policy (建立角色政策) 按鈕無法使用，請展開 Inline Policies (內嵌政策) 區域，然後選擇 click here (按一下這裡)。

5. 在 Set Permissions (設定許可) 頁面上，選擇 Custom Policy (自訂政策)，然後選擇 Select (選取)。
6. 在 Review Policy (檢閱政策) 頁面的 Policy Name (政策名稱) 欄位中，輸入用以識別此政策的名稱，例如 `CWAlarms`。
7. 將下列內容貼入至 Policy Document (政策文件) 欄位：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "cloudwatch:DescribeAlarms",
            "Resource": "*"
        }
    ]
}
```

8. 選擇 Apply Policy (套用政策)

使用 Amazon CloudWatch 事件監控部署

您可以使用 Amazon E CloudWatch vents 偵測執行個體或 CodeDeploy 作業中部署狀態 (「事件」) 的變更並做出回應。然後，根據您建立的規則，當部署或執行個體進入您在規則中指定的狀態時，E CloudWatch vents 會叫用一或多個目標動作。根據狀態變更的類型，建議您傳送通知、擷取狀態資訊、採取修正動作、啟動事件，或採取其他動作。在 CodeDeploy 作業中使用「 CloudWatch 事件」時，您可以選取下列類型的目標：

- AWS Lambda 函數
- Kinesis 串流
- Amazon SQS 佇列
- 內建目標 (`EC2 CreateSnapshot API call`、`EC2 RebootInstances API call`、`EC2 StopInstances API call`、和`EC2 TerminateInstances API call`)
- Amazon SNS 主題

下列為若干使用案例：

- 當部署失敗時，使用 Lambda 函數傳送通知到 Slack 通道。
- 推送部署或執行個體的資料到 Kinesis 串流以支援完整且即時的狀態監控。
- 當您指定的部署或執行個體事件發生時，使用 CloudWatch 警示動作自動停止、終止、重新開機或復原 Amazon EC2 執行個體。

本主題的其餘部分說明建立 CloudWatch 事件規則的基本程序 CodeDeploy。但是，在建立用於 CodeDeploy 作業的事件規則之前，您應該執行下列動作：

- 完成「 CloudWatch 事件」先決條件。如需詳細資訊，請參閱 Amazon CloudWatch 事件先決條件
- 熟悉事件中 CloudWatch 的事件、規則和目標。如需詳細資訊，請參閱什麼是 Amazon CloudWatch 活動？ 和新 CloudWatch 事件 — 追蹤並回應 AWS 資源的變更。
- 建立您將在事件規則中使用的一或多個目標。

若要為下列項目建立 CloudWatch 事件規則 CodeDeploy：

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Events (事件)。
3. 選擇 Create rule (建立規則)，然後在 Event selector (事件選擇器) 下，選擇 AWS CodeDeploy。
4. 指定一種詳細資訊類型：
   - 若要建立應用在執行個體與部署的所有變更狀態的規則，請選擇 Any detail type (任何詳細資訊型態)，然後跳到步驟六。
   - 若要建立僅套用至執行環境的規則，請選擇 [特定詳細資料類型]，然後選擇 [執行CodeDeploy 處理狀態-變更通知]。
   - 若要建立僅套用至部署的規則，請選擇 [特定詳細資料類型]，然後選擇 [CodeDeploy 部署狀態-變更通知]。
5. 指定規則套用至的狀態變更：
   - 若要建立應用至所有狀態變更的規則，請選擇 Any state (任何狀態)。
   - 若要建立僅應用至部分狀態變更的規則，請選擇 Specific state(s) (特定狀態)，然後從清單中選擇一個或多個狀態值。您可以選擇下表列出的狀態值，：

部署狀態值	執行個體狀態值
失敗	失敗
開始	開始
停止	就緒
QUEUED	成功
就緒	
成功	

6. 指定要套用規則的 CodeDeploy 應用程式：
   - 若要建立應用至所有應用程式的規則，請選擇 Any application (任何應用程式)，然後跳至步驟八。
   - 若要建立僅套用到單一個應用程式的規則，請選擇 Specific application (指定應用程式)，然後從清單中選擇此應用程式。
7. 指定部署群組的規則應用在：
   - 若要建立應用在與選定的應用程式相關之所有部署群組的規則，請選擇 Any deployment group (任何部署群組)。
   - 若要建立僅應用在與選定的應用程式相關的單一部署群組，請選擇 Specific deployment group(s) (特定部署群組)，然後從清單選擇此部署群組。
8. 檢閱您的規則設定，確定其符合您的事件監控要求。
9. 在 Targets (目標) 區域中選擇 Add target* (新增目標*)。
10. 在 Select target type (選擇目標類型) 清單中，選擇您準備好使用此規則的目標類型，然後設定此類型所需的任何其他選項。
11. 選擇設定詳細資訊。
12. 在 Configure rule details (設定規則詳細資訊) 頁面上，輸入規則的名稱和描述，然後選取 State (狀態) 方塊啟用規則。
13. 如果您對此規則感到滿意，請選擇 Create rule (建立規則)。

監視部署 AWS CloudTrail

CodeDeploy 與這項服務整合 CloudTrail，可擷取您帳戶 CodeDeploy 中由或代表您 AWS 帳戶發出的 API 呼叫，並將日誌檔案傳送到您指定的 Amazon S3 儲存貯體。 CloudTrail從 CodeDeploy 主控台、透過 CodeDeploy 命令擷取 API 呼叫 AWS CLI，或直接從 API 擷取 CodeDeploy API 呼叫。使用收集的資訊 CloudTrail，您可以判斷向哪個要求提出 CodeDeploy、提出要求的來源 IP 位址、提出要求的人員、提出要求的時間等。若要進一步了解 CloudTrail，包括如何設定和啟用它，請參閱AWS CloudTrail 使用者指南。

CodeDeploy 中的資訊 CloudTrail

在您的 AWS 帳戶中啟用 CloudTrail 記錄時，系統會在記錄檔中追蹤對 CodeDeploy 動作進行的 API 呼叫。 CodeDeploy 記錄會與其他 AWS 服務記錄一起寫入記錄檔中。 CloudTrail 根據時間週期和檔案大小決定何時建立和寫入新檔案。

所有 CodeDeploy 動作都會記錄並記錄在AWS CodeDeploy 命令列參考和 AWS CodeDeploy API 參考中。例如，建立部署、刪除應用程式和註冊應用程式修訂的呼叫會在 CloudTrail 記錄檔中產生項目。

每個日誌項目都會包含產生要求之人員的資訊。記錄檔中的使用者識別資訊可協助您判斷要求是使用根或使用者認證、具有角色或同盟使用者的暫時安全性認證，還是由其他 AWS 服務提出要求。如需詳細資訊，請參閱CloudTrail 事件參考中的 [userIdentity] 欄位。

日誌檔案可存放於儲存貯體任意長時間，但您也可以定義 Amazon S3 生命週期規則，自動封存或刪除日誌檔案。根據預設，Amazon S3 伺服器端加密 (SSE) 是用來加密您的日誌檔。

交付新的日誌檔時，您可以 CloudTrail 發佈 Amazon SNS 通知。如需詳細資訊，請參閱為 CloudTrail.

您也可以將來自多個 AWS 區域和多個 AWS 帳戶的 CodeDeploy 日誌檔彙總到單一 Amazon S3 儲存貯體。如需詳細資訊，請參閱從多個區域接收 CloudTrail 記錄檔。

瞭解 CodeDeploy 記錄檔項目

CloudTrail 記錄檔可以包含一或多個記錄項目，其中每個項目由多個 JSON 格式的事件組成。日誌項目代表任何來源提出的單一要求，並且包含所要求動作、任何參數、動作日期和時間等等的資訊。日誌項目並不保證按照任何特定的順序。也就是，日誌項目並非公用 API 呼叫的已排序堆疊追蹤。

下列範例顯示示範 CodeDeploy 建立部署群組動作的 CloudTrail 記錄項目：

```
{
 "Records": [{
```

```
  "eventVersion": "1.02",
  "userIdentity": {
   "type": "AssumedRole",
   "principalId": "AKIAI44QH8DHBEXAMPLE:203.0.113.11",
   "arn": "arn:aws:sts::123456789012:assumed-role/example-role/203.0.113.11",
   "accountId": "123456789012",
   "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
   "sessionContext": {
    "attributes": {
     "mfaAuthenticated": "false",
     "creationDate": "2014-11-27T03:57:36Z"
    },
    "sessionIssuer": {
     "type": "Role",
     "principalId": "AKIAI44QH8DHBEXAMPLE",
     "arn": "arn:aws:iam::123456789012:role/example-role",
     "accountId": "123456789012",
     "userName": "example-role"
    }
   }
  },
  "eventTime": "2014-11-27T03:57:36Z",
  "eventSource": "codedeploy.amazonaws.com",
  "eventName": "CreateDeploymentGroup",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.11",
  "userAgent": "example-user-agent-string",
  "requestParameters": {
   "applicationName": "ExampleApplication",
   "serviceRoleArn": "arn:aws:iam::123456789012:role/example-instance-group-role",
   "deploymentGroupName": "ExampleDeploymentGroup",
   "ec2TagFilters": [{
               "value": "CodeDeployDemo",
    "type": "KEY_AND_VALUE",
    "key": "Name"
           }],
           "deploymentConfigName": "CodeDeployDefault.HalfAtATime"
  },
  "responseElements": {
   "deploymentGroupId": "7d64e680-e6f4-4c07-b10a-9e117EXAMPLE"
  },
  "requestID": "86168559-75e9-11e4-8cf8-75d18EXAMPLE",
  "eventID": "832b82d5-d474-44e8-a51d-093ccEXAMPLE",
  "eventType": "AwsApiCall",
```

```
  "recipientAccountId": "123456789012"
 },
    ... additional entries ...
    ]
}
```

使用 Amazon SNS 事件通知監控部署

您可以將觸發器新增至 CodeDeploy 部署群組，以接收與該部署群組中部署或執行個體相關事件的通知。這些通知會傳送給訂閱您已參與觸發器動作一部分之 Amazon SNS 主題的收件者。

您可以接收 SMS 訊息或電子郵件訊息中的 CodeDeploy 事件通知。您也可以使用以其他方式發生指定事件時建立的 JSON 資料，例如將訊息傳送至 Amazon SQS 佇列或叫用中的函數。 AWS Lambda 如需查看 JSON 資料的結構以用於部署和執行個體觸發的詳細資訊，請參閱 CodeDeploy 觸發程序的 JSON 資料格式。

您也可選擇使用觸發程序來接收通知：

- 您是開發人員需要知道部署發生故障或停止，才能以此進行故障診斷。
- 您是系統管理員，需要知道有多少執行個體故障，才能監控 Amazon EC2 叢集的運作狀態。
- 您是想要部署和執行個體事件 at-a-glance 計數的管理員，您可以透過篩選規則取得這些規則，將不同類型的通知路由傳送到桌面電子郵件用戶端中的資料夾中。

您最多可以為每個 CodeDeploy 部署群組建立 10 個觸發器，針對下列任何一種事件類型。

部署事件	執行個體事件
• Success (成功) • 失敗 • 已開始 • 已停止 • 轉返 • 準備好 [1] • 所有部署事件	• Success (成功) • 失敗 • 已開始 • 準備好 [1] • 所有執行個體事件

部署事件	執行個體事件

[1] 僅適用於藍色/綠色部署。指出最新的應用程式修訂版以安裝在取代環境的執行個體上，並從原始環境中分出流量，現在可在負載平衡器後方重新路由。如需更多資訊，請參閱使用中的部署 CodeDeploy。

主題

將 Amazon SNS 許可授與 CodeDeploy 服務角色

觸發器才能產生通知，您在 CodeDeploy 操作中使用的服務角色必須先獲得存取 Amazon SNS 資源的權限。

將 Amazon SNS 許可授與服務角色

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在 IAM 主控台的導覽窗格中，選擇 [角色]。
3. 選擇您在 AWS CodeDeploy 操作中使用的服務角色名稱。
4. 在 Permissions (許可) 標籤的 Inline Policies (內嵌政策) 區域中，選擇 Create Role Policy (建立角色政策)。

   — 或 —

   若 Create Role Policy (建立角色政策) 按鈕無法使用，請展開 Inline Policies (內嵌政策) 區域，然後選擇 click here (按一下這裡)。
5. 在 Set Permissions (設定許可) 頁面上，選擇 Custom Policy (自訂政策)，然後選擇 Select (選取)。
6. 在 Review Policy (檢閱政策) 頁面上，Policy Name (政策名稱) 欄中輸入識別此政策的名稱，例如 `SNSPublish`。

7. 將下列內容貼入至 Policy Document (政策文件) 欄位：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sns:Publish",
            "Resource": "*"
        }
    ]
}
```

8. 選擇 Apply Policy (套用政策)

建立 CodeDeploy 事件的觸發器

您可以建立觸發器，針對 AWS CodeDeploy 部署或執行個體事件發佈 Amazon 簡單通知服務 (Amazon SNS) 主題。然後，當該事件發生時，相關聯主題的所有訂閱者都會透過主題中指定的端點 (例如 SMS 訊息或電子郵件訊息) 接收通知。Amazon SNS 提供多種訂閱主題的方法。

在建立觸發器之前，您必須設定觸發器指向的 Amazon SNS 主題。如需相關資訊，請參閱建立主題。建立主題時，建議您以`Topic-group-us-west-3-deploy-fail`或等格式為其指定一個可識別其用途的名稱`Topic-group-project-2-instance-stop`。

您還必須先將 Amazon SNS 許可授與 CodeDeploy 服務角色，才能針對觸發器傳送通知。如需相關資訊，請參閱將 Amazon SNS 許可授與 CodeDeploy 服務角色。

在您建立主題後，可以開始新增訂閱者。如需建立、管理和訂閱主題的相關資訊，請參閱什麼是 Amazon 簡單通知服務。

建立觸發器以傳送 CodeDeploy 事件通知 (主控台)

您可以使用 CodeDeploy 控制台為 CodeDeploy 事件創建觸發器。在設定程序結束時，系統會傳送測試通知訊息，藉此確認是否正確設定許可與觸發條件的詳細資訊。

若要建立 CodeDeploy 事件的觸發器

1. 在中 AWS Management Console，開啟主 AWS CodeDeploy 控台。

2. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您設定的相同使用者登入開始使用 CodeDeploy。

3. 在瀏覽窗格中，展開 [部署]，然後選擇 [應用程式]。
4. 在 Applications (應用程式) 頁面上，選擇與您要新增觸發之部署群組建立關聯的應用程式名稱。
5. 在 Application details (應用程式詳細資訊) 頁面上，選擇您要新增觸發的部署群組。
6. 選擇編輯。
7. 展開 Advanced - optional (進階 - 選用)。
8. 在 Triggers (觸發) 區域中，選擇 Create trigger (建立觸發)。
9. 在 Create deployment trigger (建立部署觸發) 窗格中，執行下列動作：

   a. 在 Trigger name (觸發名稱) 中，輸入可輕鬆識別用途的觸發名稱。建議使用 `Trigger-group-us-west-3-deploy-fail` 或 `Trigger-group-eu-central-instance-stop` 這類格式。

   b. 在事件中，選擇要觸發 Amazon SNS 主題以傳送通知的事件類型或類型。

   c. 在 Amazon SNS 主題中，選擇您為傳送此觸發器通知所建立的主題名稱。

   d. 選擇建立觸發程式。 CodeDeploy 傳送測試通知，以確認您已正確設定 CodeDeploy 和 Amazon SNS 主題之間的存取權限。根據您針對主題所選取的端點類型，且您已訂閱該主題時，會在 SMS 訊息或電子郵件訊息中收到確認訊息。

10. 選擇儲存變更。

建立觸發器以傳送 CodeDeploy 事件通知 (CLI)

您可以使用 CLI，在建立部署群組時包含觸發，也可以將觸發新增至現有部署群組。

建立觸發以傳送新部署群組的通知

建立 JSON 檔案以設定部署群組，然後使用`--cli-input-json`選項執行create-deployment-group命令。

建立 JSON 檔案的最簡單方法是使用 `--generate-cli-skeleton` 選項取得 JSON 格式的複本，然後使用純文字編輯器提供必要值。

1. 執行下列命令，然後將結果複製至純文字編輯器。

```
aws deploy create-deployment-group --generate-cli-skeleton
```

2. 將現有 CodeDeploy 應用程序的名稱添加到輸出中：

```
{
    "applicationName": "TestApp-us-east-2",
    "deploymentGroupName": "",
    "deploymentConfigName": "",
    "ec2TagFilters": [
        {
            "Key": "",
            "Value": "",
            "Type": ""
        }
    ],
    "onPremisesInstanceTagFilters": [
        {
            "Key": "",
            "Value": "",
            "Type": ""
        }
    ],
    "autoScalingGroups": [
        ""
    ],
    "serviceRoleArn": "",
    "triggerConfigurations": [
        {
            "triggerName": "",
            "triggerTargetArn": "",
            "triggerEvents": [
                ""
            ]
        }
    ]
}
```

3. 提供您要設定之參數的值。

使用指create-deployment-group令時，您必須至少提供下列參數的值：

- `applicationName`：已在您帳戶中建立的應用程式名稱。
- `deploymentGroupName`：您將建立的部署群組名稱。
- `serviceRoleArn`：在您的帳戶 CodeDeploy 中設定的現有服務角色的 ARN。如需相關資訊，請參閱步驟 2：建立服務角色 CodeDeploy。

在 `triggerConfigurations` 區段中，提供下列參數的值：

- `triggerName`：為您的觸發名稱命名，方便您識別。建議使用 `Trigger-group-us-west-3-deploy-fail` 或 `Trigger-group-eu-central-instance-stop` 這類格式。
- `triggerTargetArn`：您建立用來與觸發器產生關聯的 Amazon SNS 主題的 ARN，格式如下：`arn:aws:sns:us-east-2:444455556666:NewTestTopic`。
- `triggerEvents`：您要觸發通知的一或多種事件類型。您可以指定一或多種事件類型，以逗號分隔多個事件類型名稱 (例如，`"triggerEvents": ["DeploymentSuccess","DeploymentFailure","InstanceFailure"]`)。當您新增多種事件類型時，所有這些類型的通知都會傳送至您指定的主題，而不是每種類型的不同主題。您可以從下列事件類型來選擇：
  - DeploymentStart
  - DeploymentSuccess
  - DeploymentFailure
  - DeploymentStop
  - DeploymentRollback
  - DeploymentReady (僅適用於藍/綠部署中的取代執行個體)
  - InstanceStart
  - InstanceSuccess
  - InstanceFailure
  - InstanceReady (僅適用於藍/綠部署中的取代執行個體)

下列組態範例會針對名為 `TestApp-us-east-2` 的應用程式建立名為 `dep-group-ghi-789-2` 的部署群組，以及只要部署開始、成功或失敗就提示傳送通知的觸發：

```
{
    "applicationName": "TestApp-us-east-2",
    "deploymentConfigName": "CodeDeployDefault.OneAtATime",
```

```
    "deploymentGroupName": "dep-group-ghi-789-2",
    "ec2TagFilters": [
        {
            "Key": "Name",
            "Value": "Project-ABC",
            "Type": "KEY_AND_VALUE"
        }
    ],
    "serviceRoleArn": "arn:aws:iam::444455556666:role/AnyCompany-service-role",
    "triggerConfigurations": [
        {
            "triggerName": "Trigger-group-us-east-2",
            "triggerTargetArn": "arn:aws:sns:us-east-2:444455556666:us-east-
deployments",
            "triggerEvents": [
                "DeploymentStart",
                "DeploymentSuccess",
                "DeploymentFailure"
            ]
        }
    ]
}
```

4. 將更新儲存為 JSON 檔案，然後在您執行 create-deployment-group 命令時，使用 `--cli-input-json` 選項呼叫該檔案：

> **Important**
>
> 請確認在檔案名稱之前包含 `file://`。這是此命令必要項目。

```
aws deploy create-deployment-group --cli-input-json file://filename.json
```

在建立程序結束時，您會收到測試通知訊息，指出同時正確設定許可和觸發詳細資訊。

建立觸發以傳送現有部署群組的通知

若要使用 AWS CLI 將 CodeDeploy 事件觸發程序新增至現有部署群組，請建立 JSON 檔案以更新部署群組，然後使用`--cli-input-json`選項執行update-deployment-group命令。

建立 JSON 檔案的最簡單方法是執行 get-deployment-group 命令，取得 JSON 格式的部署群組組態複本，然後使用純文字編輯器更新參數值。

1. 執行下列命令，然後將結果複製至純文字編輯器。

```
aws deploy get-deployment-group --application-name application --deployment-group-name deployment-group
```

2. 刪除輸出中的下列內容：

   - 在輸出的開頭，刪除 `{ "deploymentGroupInfo":`。
   - 在輸出的結尾，刪除 `}`。
   - 刪除含有 `deploymentGroupId` 的資料列。
   - 刪除含有 `deploymentGroupName` 的資料列。

   文字檔案的內容現在應該與下面類似：

```
{
    "applicationName": "TestApp-us-east-2",
    "deploymentConfigName": "CodeDeployDefault.OneAtATime",
    "autoScalingGroups": [],
    "ec2TagFilters": [
        {
            "Type": "KEY_AND_VALUE",
            "Value": "Project-ABC",
            "Key": "Name"
        }
    ],
    "triggerConfigurations": [],
    "serviceRoleArn": "arn:aws:iam::444455556666:role/AnyCompany-service-role",
    "onPremisesInstanceTagFilters": []
}
```

3. 在 `triggerConfigurations` 區段中，新增 `triggerEvents`、`triggerTargetArn` 和 `triggerName` 參數的資料。如需有關觸發器組態參數的資訊，請參閱TriggerConfig。

   文字檔案的內容現在應該與下面類似。只要部署開始、成功或失敗，此程式碼就會提示傳送通知。

```
{
    "applicationName": "TestApp-us-east-2",
```

```
    "deploymentConfigName": "CodeDeployDefault.OneAtATime",
    "autoScalingGroups": [],
    "ec2TagFilters": [
        {
            "Type": "KEY_AND_VALUE",
            "Value": "Project-ABC",
            "Key": "Name"
        }
    ],
    "triggerConfigurations": [
        {
            "triggerEvents": [
                "DeploymentStart",
                "DeploymentSuccess",
                "DeploymentFailure"
            ],
            "triggerTargetArn": "arn:aws:sns:us-east-2:444455556666:us-east-
deployments",
            "triggerName": "Trigger-group-us-east-2"
        }
    ],
    "serviceRoleArn": "arn:aws:iam::444455556666:role/AnyCompany-service-role",
    "onPremisesInstanceTagFilters": []
}
```

4. 將更新儲存為 JSON 檔案，然後使用`--cli-input-json`選項執行update-deployment-group命令。請務必包含 `--current-deployment-group-name` 選項，並將 *`filename`* 替代為您 JSON 檔案的名稱：

> **Important**
>
> 請確認在檔案名稱之前包含 `file://`。這是此命令必要項目。

```
aws deploy update-deployment-group --current-deployment-group-name deployment-
group-name --cli-input-json file://filename.json
```

在建立程序結束時，您會收到測試通知訊息，指出同時正確設定許可和觸發詳細資訊。

編輯部 CodeDeploy 署群組中的觸發器

如果您的通知需求有變動，您可以修改觸發，不用建立新的觸發。

修改 CodeDeploy觸發器 (CLI)

若要在 AWS CLI 更新部署群組時使用變更 CodeDeploy 事件的觸發器詳細資料，請建立 JSON 檔案以定義部署群組內容的變更，然後使用`--cli-input-json`選項執行update-deployment-group命令。

建立 JSON 檔案的最簡單方法是執行 get-deployment-group 命令，取得 JSON 格式的目前部署群組詳細資訊，然後使用純文字編輯器編輯必要值。

1. 執行下列命令，並將 *application* 和 *deployment-group* 替代為您應用程式和部署群組的名稱：

   ```
   aws deploy get-deployment-group --application-name application --deployment-group-
   name deployment-group
   ```

2. 將命令的結果複製至純文字編輯器，然後刪除下列內容：

   - 在輸出的開頭，刪除 `{ "deploymentGroupInfo":`。
   - 在輸出的結尾，刪除 `}`。
   - 刪除含有 `deploymentGroupId` 的資料列。
   - 刪除含有 `deploymentGroupName` 的資料列。

   文字檔案的內容現在應該與下面類似：

   ```
   {
       "applicationName": "TestApp-us-east-2",
       "deploymentConfigName": "CodeDeployDefault.OneAtATime",
       "autoScalingGroups": [],
       "ec2TagFilters": [
           {
               "Type": "KEY_AND_VALUE",
               "Value": "East-1-Instances",
               "Key": "Name"
           }
       ],
       "triggerConfigurations": [
           {
   ```

```
            "triggerEvents": [
                "DeploymentStart",
                "DeploymentSuccess",
                "DeploymentFailure",
                "DeploymentStop"
            ],
            "triggerTargetArn": "arn:aws:sns:us-east-2:111222333444:Trigger-group-
us-east-2",
            "triggerName": "Trigger-group-us-east-2"
        }
    ],
    "serviceRoleArn": "arn:aws:iam::444455556666:role/AnyCompany-service-role",
    "onPremisesInstanceTagFilters": []
}
```

3. 視需要變更任意參數。如需有關觸發器組態參數的資訊，請參閱TriggerConfig。
4. 將更新儲存為 JSON 檔案，然後使用`--cli-input-json`選項執行update-deployment-group命令。請務必包含 `--current-deployment-group-name` 選項，並將 *`filename`* 替代為您 JSON 檔案的名稱：

> **Important**
>
> 請確認在檔案名稱之前包含 `file://`。這是此命令必要項目。

```
aws deploy update-deployment-group --current-deployment-group-name deployment-
group-name --cli-input-json file://filename.json
```

在建立程序結束時，您會收到測試通知訊息，指出同時正確設定許可和觸發詳細資訊。

從 CodeDeploy 部署群組刪除觸發器

因為每個部署群組都有 10 個觸發的限制，所以不再使用觸發時，建議您予以刪除。刪除觸發之後就無法復原，但您可以重新建立觸發。

從部署群組刪除觸發器 (主控台)

1. 請登入 AWS Management Console 並開啟 CodeDeploy 主控台，網址為 https://console.aws.amazon.com/codedeploy。

> **Note**
>
> 使用您設定的相同使用者登入開始使用 CodeDeploy。

2. 在瀏覽窗格中，展開 [部署]，然後選擇 [應用程式]。
3. 在 Applications (應用程式) 頁面上，選擇與您要刪除觸發之部署群組建立關聯的應用程式名稱。
4. 在 Application details (應用程式詳細資訊) 頁面上，選擇您要刪除觸發的部署群組。
5. 選擇編輯。
6. 展開 Advanced - optional (進階 - 選用)。
7. 在 Triggers (觸發) 區域中選擇您要刪除的觸發，然後選擇 Delete trigger (刪除觸發)。
8. 選擇儲存變更。

從部署群組 (CLI) 刪除觸發器

若要使用 CLI 刪除觸發器，請使用空白觸發器組態參數呼叫update-deployment-group命令，並指定：

- 與部署群組建立關聯的應用程式名稱。若要檢視應用程式名稱清單，請呼叫清單應用程式命令。
- 與應用程式建立關聯的部署群組名稱。若要檢視部署群組名稱清單，請呼叫命list-deployment-groups令。

例如：

```
aws deploy update-deployment-group --application-name application-name --current-
deployment-group-name deployment-group-name --trigger-configurations
```

CodeDeploy 觸發程序的 JSON 資料格式

您可以使用在自訂通知工作流程中啟動部署或執行個體的觸發器時建立的 JSON 輸出，例如將訊息傳送至 Amazon SQS 佇列或叫用中的函數。 AWS Lambda

> **Note**
>
> 本指南不會談論如何使用 JSON 設定通知。如需使用 Amazon SNS 將訊息傳送至 Amazon SQS 佇列的相關資訊，請參閱將 Amazon SNS 訊息傳送至 Amazon SQS 佇列。如需使用

Amazon SNS 呼叫 Lambda 函數的相關資訊，請參閱使用 Amazon SNS 通知叫用 Lambda 函數。

下列範例顯示可用於 CodeDeploy觸發程序的 JSON 輸出結構。

執行個體類型觸發的範例 JSON 輸出

```
{
    "region": "us-east-2",
    "accountId": "111222333444",
    "eventTriggerName": "trigger-group-us-east-instance-succeeded",
    "deploymentId": "d-75I7MBT7C",
    "instanceId": "arn:aws:ec2:us-east-2:444455556666:instance/i-496589f7",
    "lastUpdatedAt": "1446744207.564",
    "instanceStatus": "Succeeded",
    "lifecycleEvents": [
        {
            "LifecycleEvent": "ApplicationStop",
            "LifecycleEventStatus": "Succeeded",
            "StartTime": "1446744188.595",
            "EndTime": "1446744188.711"
        },
        {
            "LifecycleEvent": "BeforeInstall",
            "LifecycleEventStatus": "Succeeded",
            "StartTime": "1446744189.827",
            "EndTime": "1446744190.402"
        }
//More lifecycle events might be listed here
    ]
}
```

部署類型觸發的範例 JSON 輸出

```
{
    "region": "us-west-1",
    "accountId": "111222333444",
    "eventTriggerName": "Trigger-group-us-west-3-deploy-failed",
    "applicationName": "ProductionApp-us-west-3",
    "deploymentId": "d-75I7MBT7C",
    "deploymentGroupName": "dep-group-def-456",
```

```
    "createTime": "1446744188.595",
    "completeTime": "1446744190.402",
    "deploymentOverview": {
        "Failed": "10",
        "InProgress": "0",
        "Pending": "0",
        "Skipped": "0",
        "Succeeded": "0"
    },
    "status": "Failed",
    "errorInformation": {
        "ErrorCode": "IAM_ROLE_MISSING",
        "ErrorMessage": "IAM Role is missing for deployment group: dep-group-def-456"
    }
}
```

中的安全性 AWS CodeDeploy

雲安全 AWS 是最高的優先級。身為 AWS 客戶，您可以從資料中心和網路架構中獲益，該架構專為滿足對安全性最敏感的組織的需求而打造。

安全是 AWS 與您之間共同承擔的責任。共同責任模型 將此描述為雲端的安全和雲端內的安全：

- 雲端的安全性 — AWS 負責保護在 AWS 雲端中執行 AWS 服務的基礎架構。 AWS 還為您提供可以安全使用的服務。在 AWS 合規計畫中，第三方稽核員會定期測試並驗證我們的安全功效。若要深入瞭解適用於的規範遵循計劃 AWS CodeDeploy，請參閱合規方案的AWS 服務範圍。
- 雲端中的安全性 — 您的責任取決於您使用的 AWS 服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

本文件可協助您瞭解如何在使用時套用共同責任模型 CodeDeploy。下列主題說明如何設定 CodeDeploy 以符合安全性與合規性目標。您還將學習如何使用其他 AWS 服務來幫助您監控和保護您的 CodeDeploy 資源。

主題

資料保護 AWS CodeDeploy

AWS 共用責任模型適用於中的資料保護 AWS CodeDeploy。如此模型中所述， AWS 負責保護執行所有 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，我們建議您使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 保護 AWS 帳戶 登入資料並設定個別使用者。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 資源進行通訊。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。
- 使用 AWS 加密解決方案以及其中的所有默認安全控制 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。
- 如果您在透過命令列介面或 API 存取時需要經 AWS 過 FIPS 140-2 驗證的加密模組，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) 140-2 概觀。

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。這包括當您使用主控台、API CodeDeploy 或 AWS SDK 時 AWS 服務 使用或其他使用時。 AWS CLI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。

網際網路流量隱私權

CodeDeploy 是一種全受管部署服務，支援 EC2 執行個體、Lambda 函數、Amazon ECS 和現場部署伺服器。對於 EC2 執行個體和現場部署伺服器，主機代理程式會使 CodeDeploy 用 TLS 進行通訊。

目前，從代理程式到服務的通訊需要輸出網際網路連線，以便代理程式可以與公有 CodeDeploy 和 Amazon S3 服務端點進行通訊。在虛擬私有雲端中，您可以透過網際網路閘道、公司網路的站台對站台 VPN 連接，或直接連線來完成這項工作。

CodeDeploy 代理程式支援 HTTP 代理伺服器。

由 AWS PrivateLink提供支援的 Amazon VPC 端點可 CodeDeploy 在特定區域使用。如需詳細資訊，請參閱 搭 CodeDeploy 配 Amazon Virtual Private Cloud 使用。

> **Note**
>
> 僅當您部署到 Amazon EC2 /內部部署運算平台時，才需要 CodeDeploy 代理程式。使用 Amazon ECS 或 AWS Lambda 運算平台的部署不需要代理程式。

靜態加密

客戶代碼不會儲存在中 CodeDeploy。作為部署服務， CodeDeploy 正在向 EC2 執行個體或現場部署伺服器上執行的 CodeDeploy 代理程式傳送命令。然後， CodeDeploy 代理程式會使用 TLS 執

行命令。部署、部署組態、部署群組、應用程式和應用程式修訂的服務模型資料會儲存在 Amazon DynamoDB 中 AWS 擁有的金鑰，並使用擁有和管理的靜態加密。 CodeDeploy若要取得更多資訊，請參閱AWS 擁有的金鑰「」。

傳輸中加密

CodeDeploy 代理程式會啟動透 CodeDeploy 過連接埠 443 的所有通訊。代理程式會輪詢 CodeDeploy 並偵聽命令。 CodeDeploy 代理程式是開放原始碼。全部 service-to-service 和 client-to-service 通信在傳輸過程中使用 TLS 進行加密。這樣可以保護傳輸中的客戶資料以 CodeDeploy 及 Amazon S3 等其他服務。

加密金鑰管理

沒有加密金鑰可供您管理。服 CodeDeploy 務模型資料會使用 AWS 擁有的金鑰、擁有和管理 CodeDeploy。若要取得更多資訊，請參閱AWS 擁有的金鑰「」。

適用於 AWS CodeDeploy的 Identity and Access Management

AWS Identity and Access Management (IAM) 可協助系統管理員安全地控制 AWS 資源存取權。AWS 服務 IAM 管理員控制哪些人可以通過身份驗證 (登入) 和授權 (具有權限) 來使用 CodeDeploy 資源。IAM 是您可以使用的 AWS 服務 ，無需額外付費。

主題

- 物件
- 使用身分驗證
- 使用政策管理存取權
- 如何與 IAM AWS CodeDeploy 搭配使用
- AWS 的管理 (預先定義) 策略 CodeDeploy
- CodeDeploy AWS 受管理策略的更新
- AWS CodeDeploy 身分型政策範例
- 對 AWS CodeDeploy 身分與存取進行疑難排解
- CodeDeploy 許可參考
- 預防跨服務混淆代理人

物件

您使用 AWS Identity and Access Management (IAM) 的方式會有所不同，具體取決於您在進行的工作 CodeDeploy。

服務使用者 — 如果您使用 CodeDeploy 服務執行工作，則管理員會為您提供所需的認證和權限。當您使用更多 CodeDeploy 功能來完成工作時，您可能需要其他權限。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。如果無法存取中的圖徵 CodeDeploy，請參閱對 AWS CodeDeploy 身分與存取進行疑難排解。

服務管理員 — 如果您負責公司的 CodeDeploy 資源，您可能擁有完整的存取權 CodeDeploy。決定您的服務使用者應該存取哪些 CodeDeploy 功能和資源是您的工作。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。若要進一步瞭解貴公司如何搭配使用 IAM CodeDeploy，請參閱如何與 IAM AWS CodeDeploy 搭配使用。

IAM 管理員 — 如果您是 IAM 管理員，您可能想要瞭解如何撰寫政策來管理存取權限的詳細資訊 CodeDeploy。若要檢視可在 IAM 中使用的 CodeDeploy 基於身分的政策範例，請參閱。AWS CodeDeploy 身分型政策範例

使用身分驗證

驗證是您 AWS 使用身分認證登入的方式。您必須以 IAM 使用者身分或假設 IAM 角色進行驗證 (登入 AWS)。 AWS 帳戶根使用者

您可以使用透過 AWS 身分識別來源提供的認證，以聯合身分識別身分登入。 AWS IAM Identity Center (IAM 身分中心) 使用者、貴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識別的範例。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 AWS 用同盟存取時，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。如需有關登入的詳細資訊 AWS，請參閱《AWS 登入 使用指南》 AWS 帳戶中的如何登入您的。

如果您 AWS 以程式設計方式存取，請 AWS 提供軟體開發套件 (SDK) 和命令列介面 (CLI)，以使用您的認證以加密方式簽署要求。如果您不使用 AWS 工具，則必須自行簽署要求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱 IAM 使用者指南中的簽署 AWS API 請求。

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 建議您使用多重要素驗證 (MFA) 來增加帳戶的安全性。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 使用者指南中的在 AWS中使用多重要素驗證 (MFA)。

AWS 帳戶 根使用者

當您建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取該帳戶中的所有資源 AWS 服務和資源。此身分稱為 AWS 帳戶 root 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。

使用者和群組

IAM 使用者是您內部的身分，具 AWS 帳戶 有單一人員或應用程式的特定許可。建議您盡可能依賴暫時憑證，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，但角色僅提供暫時憑證。如需進一步了解，請參閱 IAM 使用者指南中的建立 IAM 使用者 (而非角色) 的時機。

IAM 角色

IAM 角色是您 AWS 帳戶 內部具有特定許可的身分。它類似 IAM 使用者，但不與特定的人員相關聯。您可以切換角色，在中暫時擔任 IAM 角色。 AWS Management Console 您可以透過呼叫 AWS CLI 或 AWS API 作業或使用自訂 URL 來擔任角色。如需使用角色的方法更多相關資訊，請參閱 IAM 使用者指南中的使用 IAM 角色。

使用暫時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 – 若要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 (信任的委託人) 存取您帳戶中的資源。角色是授予跨帳戶存取權的主要方式。但是，對於某些策略 AWS 服務，您可以將策略直接附加

到資源（而不是使用角色作為代理）。若要了解跨帳戶存取權角色和資源型政策間的差異，請參閱 IAM 使用者指南中的 IAM 角色與資源類型政策的差異。

- 跨服務訪問 — 有些 AWS 服務 使用其他 AWS 服務功能。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。
  - 轉寄存取工作階段 (FAS) — 當您使用 IAM 使用者或角色在中執行動作時 AWS，您會被視為主體。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 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 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。如需更多資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務。
  - 服務連結角色 — 服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。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 執行個體上執行的應用程式以及發出 AWS CLI 或 AWS API 請求的臨時登入資料。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 執行個體並提供給其所有應用程式，請建立連接至執行個體的執行個體設定檔。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暫時憑證。如需更多資訊，請參閱 IAM 使用者指南中的利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式。

若要了解是否要使用 IAM 角色或 IAM 使用者，請參閱 IAM 使用者指南中的建立 IAM 角色 (而非使用者) 的時機。

使用政策管理存取權

您可以透 AWS 過建立原則並將其附加至 AWS 身分識別或資源來控制中的存取。原則是一個物件 AWS ，當與身分識別或資源相關聯時，會定義其權限。 AWS 當主參與者 (使用者、root 使用者或角色工作階段) 提出要求時，評估這些原則。政策中的許可決定是否允許或拒絕請求。大多數原則會 AWS 以 JSON 文件的形式儲存在中。如需 JSON 政策文件結構和內容的更多相關資訊，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哪些內容。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。具有該原則的使用者可以從 AWS Management Console AWS CLI、或 AWS API 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。若要了解如何建立身分類型政策，請參閱 IAM 使用者指南中的建立 IAM 政策。

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。受管理的策略是獨立策略，您可以將其附加到您的 AWS 帳戶. 受管政策包括 AWS 受管政策和客戶管理的策略。若要了解如何在受管政策及內嵌政策間選擇，請參閱 IAM 使用者指南中的在受管政策和內嵌政策間選擇。

其他政策類型

AWS 支援其他較不常見的原則類型。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。如需許可範圍的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 實體許可範圍。
- 服務控制策略（SCP）— SCP 是 JSON 策略，用於指定中組織或組織單位（OU）的最大權限。AWS Organizations AWS Organizations 是一種用於分組和集中管理您企業擁 AWS 帳戶 有的多個服務。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 限制成員帳戶中實體的權限，包括每個 AWS 帳戶根使用者帳戶。如需組織和 SCP 的更多相關資訊，請參閱 AWS Organizations 使用者指南中的 SCP 運作方式。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。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。如需更多資訊，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。要了解如何在涉及多個政策類型時 AWS 確定是否允許請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

如何與 IAM AWS CodeDeploy 搭配使用

在您使用 IAM 管理存取權限之前 CodeDeploy，您應該瞭解哪些 IAM 功能可搭配使用 CodeDeploy。如需詳細資訊，請參閱 IAM 使用者指南中的與 IAM 搭配使用的AWS 服務。

主題

- CodeDeploy 身分型政策
- CodeDeploy 資源型政策
- 以 CodeDeploy 標籤為基礎的授權
- CodeDeploy IAM 角色

CodeDeploy 身分型政策

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。CodeDeploy 支援動作、資源和條件索引鍵。如需您在 JSON 政策中使用之元素的相關資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策元素參考資料。

動作

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。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。

JSON 政策的 `Action` 元素描述您可以用來允許或拒絕政策中存取的動作。原則動作通常與關聯的 AWS API 作業具有相同的名稱。有一些例外狀況，例如沒有相符的 API 操作的僅限許可動作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯動作的許可。

正在 CodeDeploy 使用的策略動作在動作之前使用前`codedeploy:`綴。例如，`codedeploy:GetApplication` 許可授予使用者執行 `GetApplication` 操作的許可。原則陳述式必須包含`Action`或`NotAction`元素。 CodeDeploy 定義了它自己的一組動作，描述您可以使用此服務執行的任務。

若要在單一陳述式中指定多個動作，請用逗號分隔，如下所示：

```
"Action": [
      "codedeploy:action1",
      "codedeploy:action2"
```

您也可以使用萬用字元 (*) 來指定多個動作。例如，包含以下動作以指定開頭是文字 Describe 的所有動作：

```
"Action": "ec2:Describe*"
```

如需 CodeDeploy 動作清單，請參閱《IAM 使用者指南》 AWS CodeDeploy中的定義動作。

如需列出所有 CodeDeploy API 動作及其套用之資源的表格，請參閱CodeDeploy 許可參考。

資源

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。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。

Resource JSON 政策元素可指定要套用動作的物件。陳述式必須包含 Resource 或 NotResource 元素。最佳實務是使用其 Amazon Resource Name (ARN) 來指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出操作)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

```
"Resource": "*"
```

例如，您可以使用 ARN 在陳述式中指示部署群組 (*myDeploymentGroup*)，如下所示：

```
"Resource": "arn:aws:codedeploy:us-
west-2:123456789012:deploymentgroup:myApplication/myDeploymentGroup"
```

您也可以使用萬用字元 (*) 來指定屬於某個帳戶的所有部署群組，如下所示：

```
"Resource": "arn:aws:codedeploy:us-west-2:123456789012:deploymentgroup:*"
```

若要指定所有資源，或如果 API 動作不支援 ARN，請在 Resource 元素中使用萬用字元 (*)，如下所示：

```
"Resource": "*"
```

某些 CodeDeploy API 動作可接受多個資源 (例如BatchGetDeploymentGroups)。若要在單一陳述式中指定多個資源，請用逗號分隔他們的 ARN，如下所示：

```
"Resource": ["arn1", "arn2"]
```

CodeDeploy 提供了一組操作來處理資 CodeDeploy源。如需可用操作的清單，請參閱 CodeDeploy 許可參考。

如需 CodeDeploy 資源類型及其 ARN 的清單，請參閱《IAM 使用者指南》 AWS CodeDeploy中的「定義資源」。如需可在其中指定每個資源 ARN 之動作的相關資訊，請參閱定義的動作。 AWS CodeDeploy

CodeDeploy 資源與營運

在中 CodeDeploy，主要資源是部署群組。在政策中，您使用 Amazon Resource Name (ARN) 來識別要套用政策的資源。 CodeDeploy 支援可與部署群組搭配使用的其他資源，包括應用程式、部署組態和執行個體。這些資源稱為「子資源」。這些資源和子資源各與唯一的 ARN 相關聯。如需詳細 Amazon 訊，請參閱中的 Amazon Web Services 一般參考.

資源類型	ARN 格式
部署群組	arn:aws:codedeploy: *region*:*account-id* :deploymentgroup: *application-name* /*deployment-group-name*
應用程式	arn:aws:codedeploy: *region*:*account-id* :application: *application-name*
部署組態	arn:aws:codedeploy: *region*:*account-id* :deploymentconfig: *deployment-configuration-name*
執行個體	arn:aws:codedeploy: *region*:*account-id* :instance/ *instance-ID*
所有 CodeDeploy 資源	arn:aws:codedeploy:*

資源類型	ARN 格式
指定區域中指定帳號擁有的所有 CodeDeploy 資源	`arn:aws:codedeploy:  region:account-id :*`

> **Note**
>
> 大多數服務都 AWS 將冒號 (:) 或正斜杠 (/) 視為 ARN 中的相同字符。但是，在資源模式和規則中 CodeDeploy 使用完全匹配。在建立事件模式時，請務必使用正確的 ARN 字元，使這些字元符合資源中的 ARN 語法。

條件索引鍵

CodeDeploy 不提供任何服務特定的條件金鑰，但它確實支援使用某些全域條件金鑰。如需詳細資訊，請參閱《IAM 使用者指南》中的AWS 全域條件內容金鑰。

範例

若要檢視以 CodeDeploy 身分為基礎的原則範例，請參閱。AWS CodeDeploy 身分型政策範例

CodeDeploy 資源型政策

CodeDeploy 不支援以資源為基礎的政策。若要檢視以資源為基礎的詳細政策頁面範例，請參閱使用以資源為基礎的政策。 AWS Lambda

以 CodeDeploy 標籤為基礎的授權

CodeDeploy 不支援標記資源或根據標籤控制存取。

CodeDeploy IAM 角色

IAM 角色是您 AWS 帳戶中具有特定許可的實體。

使用臨時登入資料 CodeDeploy

您可以搭配聯合使用暫時憑證、擔任 IAM 角色，或是擔任跨帳戶角色。您可以透過呼叫AssumeRole或等 AWS STS API 作業來取得臨時安全登入資料GetFederationToken。

CodeDeploy 支援使用臨時認證。

服務連結角色

CodeDeploy 不支援服務連結角色。

服務角色

此功能可讓服務代表您擔任服務角色。此角色可讓服務存取其他服務中的資源，以代表您完成動作。服務角色會顯示在您的 AWS 帳戶中，且屬於帳戶所有。這表示使用者可以變更此角色的權限。不過，這樣可能會破壞此服務的功能。

CodeDeploy 支援服務角色。

在中選擇 IAM 角色 CodeDeploy

在中建立部署群組資源時 CodeDeploy，您必須選擇允許 CodeDeploy 代表您存取 Amazon EC2 的角色。如果您先前已建立服務角色或服務連結角色，則會 CodeDeploy提供可供您選擇的角色清單。請務必選擇允許存取啟動和停止 EC2 執行個體的角色。

AWS 的管理 (預先定義) 策略 CodeDeploy

AWS 透過提供由建立和管理的獨立 IAM 政策來解決許多常見使用案例 AWS。這些 AWS受管理的原則會授與常見使用案例的權限，因此您可以避免調查需要哪些權限。如需詳細資訊，請參閱《IAM 使用者指南》中的 AWS 受管政策。

主題

- 下列項目的 AWS 受管政策清單 CodeDeploy
- CodeDeploy 受管理的策略和通知

下列項目的 AWS 受管政策清單 CodeDeploy

下列 AWS 受管理的策略 (您可以附加到帳戶中的使用者) 專用於 CodeDeploy：

- `AWSCodeDeployFullAccess`：授予對 CodeDeploy 的完整存取權。

> **Note**
>
> AWSCodeDeployFullAccess 不會對部署應用程式 (例如 Amazon EC2 和 Amazon S3) 所需的其他服務中的操作提供許可，僅授予以下特定操作 CodeDeploy。

- `AWSCodeDeployDeployerAccess`：授予註冊和部署修訂的權限。

- `AWSCodeDeployReadOnlyAccess`：授予對 CodeDeploy 的唯讀存取權。

- `AWSCodeDeployRole`：允 CodeDeploy 許：
  - 閱讀執行個體上的標籤，或透過 Amazon EC2 Auto Scaling 群組名稱識別您的 Amazon EC2 執行個體
  - 讀取、建立、更新和刪除 Amazon EC2 Auto Scaling 群組、生命週期勾點、擴展政策和暖池功能
  - 將資訊發佈到 Amazon SNS 主題
  - 檢索有關 Amazon CloudWatch 警報的信
  - 讀取和更新 Elastic Load Balancing 服務中的資源

  該策略包含以下代碼：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:CompleteLifecycleAction",
        "autoscaling:DeleteLifecycleHook",
        "autoscaling:DescribeAutoScalingGroups",
        "autoscaling:DescribeLifecycleHooks",
        "autoscaling:PutLifecycleHook",
        "autoscaling:RecordLifecycleActionHeartbeat",
        "autoscaling:CreateAutoScalingGroup",
        "autoscaling:CreateOrUpdateTags",
        "autoscaling:UpdateAutoScalingGroup",
        "autoscaling:EnableMetricsCollection",
        "autoscaling:DescribePolicies",
        "autoscaling:DescribeScheduledActions",
        "autoscaling:DescribeNotificationConfigurations",
        "autoscaling:SuspendProcesses",
        "autoscaling:ResumeProcesses",
        "autoscaling:AttachLoadBalancers",
        "autoscaling:AttachLoadBalancerTargetGroups",
        "autoscaling:PutScalingPolicy",
```

```
        "autoscaling:PutScheduledUpdateGroupAction",
        "autoscaling:PutNotificationConfiguration",
        "autoscaling:DescribeScalingActivities",
        "autoscaling:DeleteAutoScalingGroup",
        "autoscaling:PutWarmPool",
        "ec2:DescribeInstances",
        "ec2:DescribeInstanceStatus",
        "ec2:TerminateInstances",
        "tag:GetResources",
        "sns:Publish",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:PutMetricAlarm",
        "elasticloadbalancing:DescribeLoadBalancers",
        "elasticloadbalancing:DescribeLoadBalancerAttributes",
        "elasticloadbalancing:DescribeInstanceHealth",
        "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
        "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
        "elasticloadbalancing:DescribeTargetGroups",
        "elasticloadbalancing:DescribeTargetGroupAttributes",
        "elasticloadbalancing:DescribeTargetHealth",
        "elasticloadbalancing:RegisterTargets",
        "elasticloadbalancing:DeregisterTargets"
      ],
      "Resource": "*"
    }
  ]
}
```

- `AWSCodeDeployRoleForLambda`：授 CodeDeploy予部署所需的存取權 AWS Lambda 和任何其他資源的權限。

- `AWSCodeDeployRoleForECS`：授 CodeDeploy 予存取 Amazon ECS 和部署所需任何其他資源的權限。

- `AWSCodeDeployRoleForECSLimited`：授 CodeDeploy予存取 Amazon ECS 和部署所需的任何其他資源的權限，但下列例外：
  - 在 AppSpec 檔案的`hooks`區段中，只能使用名稱開頭為的 `CodeDeployHook_` Lambda 函數。如需詳細資訊，請參閱 AppSpec Amazon ECS 部署的「掛鉤」部分。

  - S3 儲存貯體的存取權限制為具有註冊標籤 `UseWithCodeDeploy` 且值為 `true` 的 S3 儲存貯體。如需詳細資訊，請參閱物件標籤。

- `AmazonEC2RoleforAWSCodeDeployLimited`：授 CodeDeploy 予取得和列出 CodeDeploy Amazon S3 儲存貯體中物件的權限。該策略包含以下代碼：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::*/CodeDeploy/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/UseWithCodeDeploy": "true"
                }
            }
        }
    ]
}
```

部署程序某些層面的權限會授與其他兩種代表作業的角色類型 CodeDeploy：

- IAM 執行個體設定檔是您附加到 Amazon EC2 執行個體的 IAM 角色。此設定檔包括存取存放應用程式的 Amazon S3 儲存貯體或存 GitHub 放庫所需的許可。如需詳細資訊，請參閱 步驟 4：為您的 Amazon EC2 執行個體建立 IAM 執行個體設定檔。

- 服務角色是一種 IAM 角色，可授予 AWS 服務許可，以便它可以存取 AWS 資源。您附加至服務角色的原則會決定服務可存取哪些 AWS 資源，以及可以對這些資源執行的動作。對於 CodeDeploy，服務角色可用於下列項目：
  - 讀取套用至執行個體的標籤或與執行個體相關聯的 Amazon EC2 Auto Scaling 群組名稱。這可 CodeDeploy 以識別它可以部署應用程式的執行個體。
  - 在執行個體、Amazon EC2 Auto Scaling 群組和 Elastic Load Balancing 負載平衡器上執行操作。
  - 將資訊發佈到 Amazon SNS 主題，以便在發生指定的部署或執行個體事件時傳送通知。
  - 擷取 CloudWatch 警示的相關資訊，以針對部署設定警示監控。

  如需詳細資訊，請參閱 步驟 2：建立服務角色 CodeDeploy。

您也可以建立自訂 IAM 政策，以授予 CodeDeploy動作和資源的許可。您可以將這些自訂政策附加到 IAM 角色，然後將角色指派給需要權限的使用者或群組。

CodeDeploy 受管理的策略和通知

CodeDeploy 支援通知，可通知使用者部署的重要變更。 CodeDeploy 包含通知功能的政策聲明的受管理策略。如需詳細資訊，請參閱什麼是通知？。

完整存取受管政策中的通知相關許可

`AWSCodeDeployFullAccess` 受管政策包含下列陳述式，允許對通知的完整存取權限。套用此受管政策的使用者也可以針對通知建立和管理 Amazon SNS 主題、訂閱和取消訂閱使用者主題、列出要選擇作為通知規則目標的主題，以及列出針對 Slack 設定的用 AWS Chatbot 戶端。

```
    {
        "Sid": "CodeStarNotificationsReadWriteAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:CreateNotificationRule",
            "codestar-notifications:DescribeNotificationRule",
            "codestar-notifications:UpdateNotificationRule",
            "codestar-notifications:DeleteNotificationRule",
            "codestar-notifications:Subscribe",
            "codestar-notifications:Unsubscribe"
        ],
        "Resource": "*",
        "Condition" : {
            "StringLike" : {"codestar-notifications:NotificationsForResource" :
 "arn:aws:codedeploy:*"}
```

```
        }
    },
    {
        "Sid": "CodeStarNotificationsListAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:ListNotificationRules",
            "codestar-notifications:ListTargets",
            "codestar-notifications:ListTagsforResource",
            "codestar-notifications:ListEventTypes"
        ],
        "Resource": "*"
    },
    {
        "Sid": "CodeStarNotificationsSNSTopicCreateAccess",
        "Effect": "Allow",
        "Action": [
            "sns:CreateTopic",
            "sns:SetTopicAttributes"
        ],
        "Resource": "arn:aws:sns:*:*:codestar-notifications*"
    },
    {
        "Sid": "SNSTopicListAccess",
        "Effect": "Allow",
        "Action": [
            "sns:ListTopics"
        ],
        "Resource": "*"
    },
    {
        "Sid": "CodeStarNotificationsChatbotAccess",
        "Effect": "Allow",
        "Action": [
            "chatbot:DescribeSlackChannelConfigurations",
            "chatbot:ListMicrosoftTeamsChannelConfigurations"
          ],
       "Resource": "*"
    }
```

唯讀受管政策中的通知相關許可

AWSCodeDeployReadOnlyAccess 受管政策包含下列陳述式，允許對通知的唯讀存取權限。套用此政策的使用者可以檢視資源的通知，但無法建立、管理或訂閱通知。

```
  {
       "Sid": "CodeStarNotificationsPowerUserAccess",
       "Effect": "Allow",
       "Action": [
           "codestar-notifications:DescribeNotificationRule"
       ],
       "Resource": "*",
       "Condition" : {
           "StringLike" : {"codestar-notifications:NotificationsForResource" :
"arn:aws:codedeploy:*"}
       }
   },
   {
       "Sid": "CodeStarNotificationsListAccess",
       "Effect": "Allow",
       "Action": [
           "codestar-notifications:ListNotificationRules",
           "codestar-notifications:ListEventTypes",
           "codestar-notifications:ListTargets"
       ],
       "Resource": "*"
   }
```

如需 IAM 和通知的詳細資訊，請參閱通知的 Identity and Access Management。AWS CodeStar

CodeDeploy AWS 受管理策略的更新

檢視 CodeDeploy 自此服務開始追蹤這些變更以來的 AWS 受管理策略更新詳細資料。如需有關此頁面變更的自動警示，請訂閱上的 RSS 摘要 CodeDeploy 文件歷史紀錄。

變更	描述	日期
AWSCodeDeployRole 受管理策略 — 現有策略的更新	已將elasticloadbalancing:DescribeLoadBalancerAttributes 和動elasticloadbalanci	2023 年 8 月 16 日

變更	描述	日期
	ng:DescribeTargetGroupAttributes 作新增至原則陳述式，以支援 Elastic Load Balancing 變更。 如需此原則的詳細資訊，請參閱AWSCodeDeployRole。	
AWSCodeDeployFullAccess 受管理策略 — 現有策略的更新	已將chatbot:ListMicrosoftTeamsChannelConfigurations 動作新增至政策聲明以支援通知變更。 如需此原則的詳細資訊，請參閱AWSCodeDeployRole。	2023 年 5 月 11 日
AWSCodeDeployRole 受管理策略 — 現有策略的更新	已在政策聲明中新增 autoscaling:CreateOrUpdateTags 動作，以支援 Amazon EC2 Auto Scaling 授權變更。 如需此原則的詳細資訊，請參閱AWSCodeDeployRole。	2023 年 2 月 3 日
AmazonEC2RoleforAWSCodeDeployLimited 受管理策略 — 現有策略的更新	已從包含s3:ExistingObjectTag/UseWithCodeDeploy 條件的政策聲明中移除s3:ListBucket 動作。 如需此原則的詳細資訊，請參閱AmazonEC2RoleforAWSCodeDeployLimited。	2021 年 11 月 22 日

變更	描述	日期
AWSCodeDeployRole 受管理策略 — 現有策略的更新	已新增autoscaling:PutWarmPool 動作，以支援將暖集區新增至 Amazon EC2 Auto Scaling 群組以進行藍色/綠色部署。 移除不必要的重複動作。	2021 年 5 月 18 日
CodeDeploy 開始追蹤變更	CodeDeploy 開始追蹤其 AWS 受管理策略的變更。	2021 年 5 月 18 日

AWS CodeDeploy 身分型政策範例

依預設，使用者沒有建立或修改 CodeDeploy資源的權限。他們也無法使用 AWS Management Console AWS CLI、或 AWS API 執行工作。您必須建立 IAM 政策以授予 IAM 角色權限，才能在所需的指定資源上執行 API 操作。然後，您必須將這些 IAM 角色附加到需要這些許可的使用者或群組。

若要了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的在 JSON 標籤上建立政策。

在中 CodeDeploy，以身分識別為基礎的原則可用來管理與部署程序相關之各種資源的權限。您可以控制下列資源類型的存取：

- 應用程式與應用程式修訂。
- 部署。
- 部署組態。
- 執行個體與內部部署執行個體。

資源政策控制的功能會因資源類型而有所不同，如下表所述：

資源類型	功能
全部	檢視及列出資源的詳細資訊
應用程式	建立 資源

資源類型	功能
部署組態 部署群組	刪除資源
部署	建立部署 停止部署
應用程式修訂	註冊應用程式修訂
應用程式 部署群組	更新資源
現場部署執行個體	為執行個體新增標籤 從執行個體移除標籤 註冊執行個體 取消註冊執行個體

以下範例顯示的許可政策，允許使用者刪除 **us-west-2** 區域內與名為 **WordPress_App** 應用程式建立關聯，且名為 **WordPress_DepGroup** 的部署群組。

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:DeleteDeploymentGroup"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:444455556666:deploymentgroup:WordPress_App/
WordPress_DepGroup"
      ]
    }
  ]
```

```
}
```

主題

客戶受管政策範例

在本節中，您可以找到授與各種 CodeDeploy 動作權限的範例原則。當您使用 CodeDeploy API、AWS SDK 或. AWS CLI您必須為在主控台中執行的動作授與其他許可。若要進一步了解如何授與主控台許可，請參閱 使用 CodeDeploy 主控台。

> **Note**
>
> 所有範例皆使用美國西部 (奧勒岡) 區域 (us-west-2) 及虛構帳戶 ID。

範例

範例 1：允許在單一區域 CodeDeploy 執行作業的權限

下列範例僅授與在**us-west-2**區域中 CodeDeploy 執行作業的權限：

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:*"
      ],
```

```
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:444455556666:*"
      ]
    }
  ]
}
```

範例 2：允許註冊單一應用程式修訂的權限

以下範例會授予許可，註冊 **us-west-2** 區域內所有開頭為 **Test** 應用程式的應用程式修訂：

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:RegisterApplicationRevision"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:444455556666:application:Test*"
      ]
    }
  ]
}
```

範例 3：允許為單一部署群組建立部署的權限

下列範例允許權限為與名稱為的應用程式**WordPress_DepGroup**相關聯的部署群組、名為的自訂部署組態**WordPress_App**，以及與名稱**ThreeQuartersHealthy**為的應用程式相關聯的任何應用程式修訂版本建立部署**WordPress_App**。所有這些資源都位於 **us-west-2** 區域。

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:CreateDeployment"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:444455556666:deploymentgroup:WordPress_App/
WordPress_DepGroup"
```

```
      ]
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:GetDeploymentConfig"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-
west-2:444455556666:deploymentconfig:ThreeQuartersHealthy"
      ]
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:GetApplicationRevision"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:444455556666:application:WordPress_App"
      ]
    }
  ]
}
```

政策最佳實務

以身分識別為基礎的政策會決定某人是否可以建立、存取或刪除您帳戶中的 CodeDeploy 資源。這些動作可能會讓您的 AWS 帳戶產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管原則並邁向最低權限權限 — 若要開始授與使用者和工作負載的權限，請使用可授與許多常見使用案例權限的AWS 受管理原則。它們可用在您的 AWS 帳戶. 建議您透過定義特定於您使用案例的 AWS 客戶管理政策，進一步降低使用權限。如需更多資訊，請參閱 IAM 使用者指南中的 AWS 受管政策或任務職能的AWS 受管政策。
- 套用最低許可許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的權限。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 中的政策和許可。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。您也可以使用條件來授與對服務動作的存取權 (如透過特定) 使用這些動作 AWS 服務，例如 AWS CloudFormation。如需更多資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策元素：條件。

- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您編寫安全且實用的政策。如需更多資訊，請參閱 IAM 使用者指南中的 IAM Access Analyzer 政策驗證。
- 需要多因素身份驗證 (MFA) — 如果您的案例需要 IAM 使用者或根使用者 AWS 帳戶，請開啟 MFA 以獲得額外的安全性。若要在呼叫 API 作業時請求 MFA，請將 MFA 條件新增至您的政策。如需更多資訊，請參閱 IAM 使用者指南中的設定 MFA 保護的 API 存取。

如需 IAM 中最佳實務的相關資訊，請參閱 IAM 使用者指南中的 IAM 安全最佳實務。

使用 CodeDeploy 主控台

如果您使用 CodeDeploy 主控台，您必須擁有一組最低權限，才能說明 AWS 帳戶的其他 AWS 資源。若要 CodeDeploy 在 CodeDeploy 主控台中使用，您必須具有下列服務的權限：

- Amazon EC2 Auto Scaling
- AWS CodeDeploy
- Amazon Elastic Compute Cloud
- Elastic Load Balancing
- AWS Identity and Access Management
- Amazon Simple Storage Service
- Amazon Simple Notification Service
- Amazon CloudWatch

如果您建立的 IAM 政策限制比最低所需許可更嚴格，則主控台將無法針對具有該 IAM 政策角色的使用者正常運作。若要確保這些使用者仍然可以使用 CodeDeploy主控台，請同時將`AWSCodeDeployReadOnlyAccess`受管理的策略附加到指派給使用者的角色，如中所述AWS 的管理 (預先定義) 策略 CodeDeploy。

對於只對 AWS CLI 或 CodeDeploy API 進行呼叫的使用者，您不需要允許最低主控台權限。

允許使用者檢視他們自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視附加到他們使用者身分的內嵌及受管政策。此原則包含在主控台上或以程式設計方式使用 AWS CLI 或 AWS API 完成此動作的權限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

對 AWS CodeDeploy 身分與存取進行疑難排解

使用下列資訊可協助您診斷和修正使用和 IAM 時可能會遇到的 CodeDeploy 常見問題。

主題

- 我沒有授權執行 iam：PassRole
- 我想允許 AWS 帳戶以外的人員存取我的 CodeDeploy資源

我沒有授權執行 iam：PassRole

如果您收到未獲授權執行`iam:PassRole`動作的錯誤訊息，則必須更新您的原則以允許您將角色傳遞給 CodeDeploy。

有些 AWS 服務 允許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。

當名為的 IAM 使用者`marymajor`嘗試使用主控台執行中的動作時，會發生下列範例錯誤 CodeDeploy。但是，動作請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 系統管理員。您的管理員提供您的簽署憑證。

我想允許 AWS 帳戶以外的人員存取我的 CodeDeploy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：

- 若要瞭解是否 CodeDeploy 支援這些功能，請參閱如何與 IAM AWS CodeDeploy 搭配使用。
- 若要了解如何提供對您所擁有資源 AWS 帳戶 的存取權，請參閱《IAM 使用者指南》中您擁有的另一 AWS 帳戶 個 IAM 使用者提供存取權限。
- 若要了解如何將資源存取權提供給第三方 AWS 帳戶，請參閱 IAM 使用者指南中的提供第三方 AWS 帳戶 擁有的存取權。
- 若要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)。
- 若要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 角色與資源型政策的差異。

CodeDeploy 許可參考

當您設定可附加至 IAM 身分 (身分型政策) 的存取和寫入許可政策時，請使用下表。此表格會列出每個 CodeDeploy API 作業、您可授與執行動作之權限的動作，以及用於授與權限的資源 ARN 格式。您可以在政策的 `Action` 欄位中指定動作。您可以使用或不使用萬用字元 (*)，指定 ARN 做為政策之 `Resource` 欄位中的資源值。

您可以在 CodeDeploy 原則中使用 AWS寬條件金鑰來表示條件。如需完整的 AWS全金鑰清單，請參閱 IAM 使用者指南中的可用金鑰。

若要指定動作，請使用 `codedeploy:` 字首，後面接著 API 操作名稱 (例如 `codedeploy:GetApplication` 和 `codedeploy:CreateApplication`)。若要在單一陳述式中指定多個動作，請用逗號加以分隔 (例如 `"Action": ["codedeploy:action1", "codedeploy:action2"]`)。

使用萬用字元

您可以在您的 ARN 中使用萬用字元 (*) 指定多個動作或資源。例如，`codedeploy:*`指定所有 CodeDeploy 動作，並`codedeploy:Get*`指定以該字開頭的所有 CodeDeploy 動作`Get`。以下範例會授予存取所有名稱開頭為 `West`，且和名稱開頭為 `Test` 應用程式建立關聯的部署群組。

```
arn:aws:codedeploy:us-west-2:444455556666:deploymentgroup:Test*/West*
```

您可以針對下表列出的資源使用萬用字元：

- *application-name*
- *deployment-group-name*
- *deployment-configuration-name*
- *instance-ID*

萬用字元無法用於 *region* 或 *account-id*。如需萬用字元的詳細資訊，請參閱《IAM 使用者指南》中的 IAM 識別符。

> **Note**
>
> 在每個動作的 ARN 中，資源後方會跟隨一個冒號 (:)。您也可以使用斜線 (/) 跟隨資源。如需詳細資訊，請參閱CodeDeploy範例 ARN。

CodeDeploy API 操作和動作所需的權限

AddTagsToOnPremisesInstances

動作：codedeploy:AddTagsToOnPremisesInstances

若要將標籤新增至一個或多個現場部署執行個體，則為必要許可。

資源：arn:aws:codedeploy:*region*:*account-id*:instance/*instance-ID*

BatchGetApplicationRevisions

動作：codedeploy:BatchGetApplicationRevisions

若要取得與 使用者相關聯的多個應用程式修訂版本相關資訊，則為必要許可。

資源： arn:aws:codedeploy:*region*:*account-id*:application:*application-name*

BatchGetApplications

動作：codedeploy:BatchGetApplications

若要取得與 使用者相關聯的多個應用程式相關資訊，則為必要許可。

資源：arn:aws:codedeploy:*region*:*account-id*:application:*

BatchGetDeploymentGroups

動作：codedeploy:BatchGetDeploymentGroups

若要取得與 使用者相關聯的多個部署群組相關資訊，則為必要許可。

資源：arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

BatchGetDeploymentInstances

動作：codedeploy:BatchGetDeploymentInstances

若要取得一個或多個執行個體 (其屬於部署群組一部分) 的相關資訊，則為必要許可。

資源：arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

BatchGetDeployments

動作：codedeploy:BatchGetDeployments

若要取得與 使用者相關聯的多個部署相關資訊，則為必要許可。

資源：arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

BatchGetOnPremisesInstances

動作：codedeploy:BatchGetOnPremisesInstances

若要取得一個或多個現場部署執行個體的相關資訊，則為必要許可。

資源：arn:aws:codedeploy:*region*:*account-id*:*

ContinueDeployment

動作：codedeploy:ContinueDeployment

藍/綠部署期間必須使用 Elastic Load Balancing 器在取代環境中註冊執行個體。

資源：arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

CreateApplication

動作：codedeploy:CreateApplication

若要建立與 使用者相關聯的應用程式，則為必要許可。

資源：arn:aws:codedeploy:*region*:*account-id*:application:*application-name*

CreateDeployment[1]

動作：codedeploy:CreateDeployment

若要建立與 使用者相關聯的應用程式部署，則為必要許可。

資源：arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

CreateDeploymentConfig

動作：codedeploy:CreateDeploymentConfig

若要建立與 使用者相關聯的自訂部署組態，則為必要許可。

資源：arn:aws:codedeploy:*region*:*account-id*:deploymentconfig/*deployment-configuration-name*

CreateDeploymentGroup

動作：codedeploy:CreateDeploymentGroup

若要建立與 使用者相關聯的應用程式部署群組，則為必要許可。

資源：arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

DeleteApplication

動作：codedeploy:DeleteApplication

若要刪除與 使用者相關聯的應用程式，則為必要許可。

資源：arn:aws:codedeploy:*region*:*account-id*:application:*application-name*

DeleteDeploymentConfig

動作：codedeploy:DeleteDeploymentConfig

若要刪除與 使用者相關聯的自訂部署組態，則為必要許可。

資源：arn:aws:codedeploy:*region*:*account-id*:deploymentconfig/*deployment-configuration-name*

DeleteDeploymentGroup

動作：codedeploy:DeleteDeploymentGroup

若要刪除與 使用者相關聯的應用程式部署群組，則為必要許可。

資源：arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

DeregisterOnPremisesInstance

動作：codedeploy:DeregisterOnPremisesInstance

若要取消註冊現場部署執行個體，則為必要許可。

資源： arn:aws:codedeploy:*region*:*account-id*:instance/*instance-ID*

GetApplication

動作：codedeploy:GetApplication

若要取得與 使用者相關聯的單一應用程式相關資訊，則為必要許可。

資源：arn:aws:codedeploy:*region*:*account-id*:application:*application-name*

GetApplicationRevision

動作：codedeploy:GetApplicationRevision

若要取得與 使用者相關聯的單一應用程式修訂版本相關資訊，則為必要許可。

資源：arn:aws:codedeploy:*region*:*account-id*:application:*application-name*

GetDeployment

動作：codedeploy:GetDeployment

若要針對與 使用者相關聯的應用程式，取得部署群組的單一部署相關資訊，則為必要許可。

資源： arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

GetDeploymentConfig

動作：codedeploy:GetDeploymentConfig

若要取得與 使用者相關聯的單一部署組態相關資訊，則為必要許可。

資源：arn:aws:codedeploy:*region*:*account-id*:deploymentconfig/*deployment-configuration-name*

GetDeploymentGroup

動作：codedeploy:GetDeploymentGroup

若要取得與 使用者相關聯的應用程式單一部署群組相關資訊，則為必要許可。

資源：arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

GetDeploymentInstance

動作：codedeploy:GetDeploymentInstance

若要取得與 使用者相關聯部署中的單一執行個體相關資訊，則為必要許可。

資源：arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

GetOnPremisesInstance

動作：codedeploy:GetOnPremisesInstance

若要取得單一現場部署執行個體的相關資訊，則為必要許可。

資源：arn:aws:codedeploy:*region*:*account-id*:instance/*instance-ID*

ListApplicationRevisions

動作：codedeploy:ListApplicationRevisions

若要取得與 使用者相關聯的所有應用程式修訂版本相關資訊，則為必要許可。

資源： arn:aws:codedeploy:*region*:*account-id*:application:*

ListApplications

動作：codedeploy:ListApplications

若要取得與 使用者相關聯的所有應用程式相關資訊，則為必要許可。

資源：arn:aws:codedeploy:*region*:*account-id*:application:*

ListDeploymentConfigs

動作：codedeploy:ListDeploymentConfigs

若要取得與 使用者相關聯的所有部署組態相關資訊，則為必要許可。

資源：arn:aws:codedeploy:*region*:*account-id*:deploymentconfig/*

ListDeploymentGroups

動作：codedeploy:ListDeploymentGroups

若要取得與 使用者相關聯的所有應用程式部署群組相關資訊，則為必要許可。

資源：arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*

ListDeploymentInstances

動作：codedeploy:ListDeploymentInstances

取得部署中與使用者相關聯之所有執行個體的相關資訊是必要的。

資源：arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

ListDeployments

動作：codedeploy:ListDeployments

若要取得與使用者相關聯之部署群組的所有部署資訊，或取得與使用者相關聯的所有部署，則為必要項目。

資源：arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

ListGitHubAccountTokenNames

動作：codedeploy:ListGitHubAccountTokenNames

需要取得 GitHub帳戶的儲存連線名稱清單。

資源：arn:aws:codedeploy:*region*:*account-id*:*

ListOnPremisesInstances

動作：codedeploy:ListOnPremisesInstances

若要取得一個或多個現場部署執行個體名稱的清單，則為必要許可。

資源：arn:aws:codedeploy:*region*:*account-id*:*

RegisterApplicationRevision

動作：codedeploy:RegisterApplicationRevision

若要註冊與 使用者相關聯的應用程式修訂版本相關資訊，則為必要許可。

資源：arn:aws:codedeploy:*region*:*account-id*:application:*application-name*

RegisterOnPremisesInstance

動作：codedeploy:RegisterOnPremisesInstance

必須使用註冊內部部署執行個體 CodeDeploy。

資源：arn:aws:codedeploy:*region*:*account-id*:instance/*instance-ID*

RemoveTagsFromOnPremisesInstances

動作：codedeploy:RemoveTagsFromOnPremisesInstances

若要從一個或多個現場部署執行個體移除標籤，則為必要許可。

資源：arn:aws:codedeploy:*region*:*account-id*:instance/*instance-ID*

SkipWaitTimeForInstanceTermination

動作：codedeploy:SkipWaitTimeForInstanceTermination

必要許可，用於在藍色/綠色部署中覆寫指定等待時間，並於流量成功路由後，立即開始終止原始環境中的執行個體。

資源：arn:aws:codedeploy:*region*:*account-id*:instance/*instance-ID*

StopDeployment

動作：codedeploy:StopDeployment

若要針對與 使用者相關聯的應用程式，停止部署群組正在進行的部署，則為必要許可。

資源：arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

UpdateApplication[3]

動作：codedeploy:UpdateApplication

若要變更與 使用者相關聯的應用程式相關資訊，則為必要許可。

資源：arn:aws:codedeploy:*region*:*account-id*:application:*application-name*

UpdateDeploymentGroup[3]

動作：codedeploy:UpdateDeploymentGroup

若要變更與 使用者相關聯的應用程式單一部署群組相關資訊，則為必要許可。

資源：arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

[1] 當您指定CreateDeployment權限時，您還必須指定部署組態的GetDeploymentConfig權限，以及/GetApplicationRevision或應用程式修訂版的RegisterApplicationRevision權限。

[2] 適用於ListDeployments您提供部署群組的時間，但在列出與使用者相關聯的所有部署時不適用。

[3] 對於UpdateApplication，您必須同時擁有舊應用程式名稱和新應用程式名稱的UpdateApplication權限。對於涉及變更部署群組名稱的 UpdateDeploymentGroup 動作，您必須同時擁有舊的及新的部署群組名稱的 UpdateDeploymentGroup 許可。

預防跨服務混淆代理人

混淆代理人問題屬於安全性問題，其中沒有執行動作許可的實體可以強制具有更多許可的實體執行該動作。在中 AWS，跨服務模擬可能會導致混淆的副問題。在某個服務 (呼叫服務) 呼叫另一個服務 (被呼叫服務) 時，可能會發生跨服務模擬。可以操縱呼叫服務來使用其許可，以其不應有存取許可的方式對其他客戶的資源採取動作。為了防止這種情況發生， AWS 提供的工具可協助您透過已授予您帳戶中資源存取權的服務主體來保護所有服務的資料。

我們建議在資源政策中使用 aws: SourceArn 和 aws: SourceAccount 全域條件上下文金鑰，以限制將另一個服務 CodeDeploy 提供給資源的許可。如果同時使用這兩個全域條件內容索引鍵，且 `aws:SourceArn` 值包含帳戶 ID，則在相同政策陳述式中使用 `aws:SourceAccount` 值和 `aws:SourceArn` 值中的帳戶時，必須使用相同的帳戶 ID。如果您想要僅允許一個資源與跨服務存取相關聯，則請使用 `aws:SourceArn`。如`aws:SourceAccount`果您希望該帳號中的任何資源與跨服務使用相關聯，請使用此選項。

對於 EC2/現場部署、 AWS Lambda 和一般 Amazon ECS 部署，其 CodeDeploy 值`aws:SourceArn`應包括允許擔任 IAM 角色的 CodeDeploy 部署群組 ARN。

對於透過建立的 Amazon ECS 藍/綠部署 AWS CloudFormation，的值`aws:SourceArn`應包含允許使用 IAM 角色的 CloudFormation CodeDeploy 堆疊 ARN。

防止混淆的副問題的最有效方法是使用帶有完整 ARN 資源的`aws:SourceArn`密鑰。如果您不知道完整的 ARN，或者您要指定多個資源，請在未知部分使用萬用字元 (*)。

例如，您可以在 EC2 /內部部署、 AWS Lambda 或一般 Amazon ECS 部署中使用下列信任政策：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "codedeploy.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "StringLike": {
```

```
                    "aws:SourceArn": "arn:aws:codedeploy:us-
east-1:111122223333:deploymentgroup:myApplication/*"
                }
            }
        }
    ]
}
```

對於透過建立的 Amazon ECS 藍/綠部署 AWS CloudFormation，您可以使用：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "codedeploy.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "StringLike": {
                    "aws:SourceArn": "arn:aws:cloudformation:us-
east-1:111122223333:stack/MyCloudFormationStackName/*"
                }
            }
        }
    ]
}
```

登錄和監控 CodeDeploy

本節提供中監視、記錄和事件回應的概觀 CodeDeploy。

稽核與之間的所有互動 CodeDeploy

CodeDeploy 與這項服務整合 AWS CloudTrail，可擷取您帳戶 CodeDeploy 中由或代表您 AWS 帳戶發出的 API 呼叫，並將日誌檔案傳送到您指定的 S3 儲存貯體。 CloudTrail 從 CodeDeploy 主控台、

透過 CodeDeploy 命令擷取 API 呼叫 AWS CLI，或直接從 API 擷取 CodeDeploy API 呼叫。使用收集的資訊 CloudTrail，您可以判斷向哪個要求提出 CodeDeploy、提出要求的來源 IP 位址、提出要求的人員、提出要求的時間等。若要進一步瞭解 CloudTrail，請參閱《使用指南》中的〈AWS CloudTrail 使用 CloudTrail記錄檔〉。

您可以透過設定 Amazon CloudWatch 代理程式在 CloudWatch 主控台中檢視彙總資料，或登入執行個體以檢視日誌檔，來檢視 CodeDeploy 部署所建立的日誌資料。如需詳細資訊，請參閱 傳送 CodeDeploy 代理程式記錄至 CloudWatch。

警示與事件管理

您可以使用 Amazon E CloudWatch vents 偵測執行個體或 CodeDeploy 操作中部署 (事件) 狀態的變更並做出回應。然後，根據您建立的規則，當部署或執行個體進入您在規則中指定的狀態時，E CloudWatch vents 會叫用一或多個目標動作。根據狀態變更的類型，建議您傳送通知、擷取狀態資訊、採取修正動作、啟動事件，或採取其他動作。當您在 CodeDeploy 作業中使用「 CloudWatch 事件」時，您可以選取下列類型的目標：

- AWS Lambda 函數
- Kinesis 串流
- Amazon SQS SQS 隊列
- 內建目標 (CloudWatch 警示動作)
- Amazon SNS 主題

下列為若干使用案例：

- 當部署失敗時，使用 Lambda 函數傳送通知到 Slack 通道。
- 推送部署或執行個體的資料到 Kinesis 串流以支援完整且即時的狀態監控。
- 當您指定的部署或執行個體事件發生時，使用 CloudWatch 警示動作自動停止、終止、重新開機或復原 EC2 執行個體。

有關更多信息，請參閱 Amazon 用 CloudWatch 戶指南中的 Amazon CloudWatch 事件是什麼。

符合性驗證 AWS CodeDeploy

若要瞭解 AWS 服務 是否屬於特定規範遵循方案的範圍內，請參閱AWS 服務 遵循規範計劃方案中的，並選擇您感興趣的合規方案。如需一般資訊，請參閱AWS 規範計劃AWS。

您可以使用下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱下載中的報告中的 AWS Artifact。

您在使用時的合規責任取決 AWS 服務 於資料的敏感性、公司的合規目標以及適用的法律和法規。AWS 提供下列資源以協助遵循法規：

- 安全性與合規性快速入門指南 — 這些部署指南討論架構考量，並提供部署以安全性和合規性 AWS 為重點的基準環境的步驟。
- 在 Amazon Web Services 上架構 HIPAA 安全性與合規性 — 本白皮書說明公司如何使用建立符合 HIPAA 資格的應 AWS 用程式。

> **Note**
>
> 並非所有人 AWS 服務 都符合 HIPAA 資格。如需詳細資訊，請參閱 HIPAA 資格服務參照。

- AWS 合規資源AWS — 此工作簿和指南集合可能適用於您的產業和所在地。
- AWS 客戶合規指南 — 透過合規的角度瞭解共同的責任模式。這份指南總結了在多個架構 (包括美國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)) 中，保 AWS 服務 護指引並對應至安全控制的最佳實務。
- 使用AWS Config 開發人員指南中的規則評估資源 — 此 AWS Config 服務會評估您的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 提供了內部安全狀態的全面視圖 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 controls reference。
- Amazon GuardDuty — 透過監控環境中的 AWS 帳戶可疑和惡意活動，藉此 AWS 服務 偵測您的工作負載、容器和資料的潛在威脅。 GuardDuty 可協助您滿足特定合規性架構所要求的入侵偵測需求，例如 PCI DSS 等各種合規性需求。
- AWS Audit Manager— 這 AWS 服務 有助於您持續稽核您的 AWS 使用情況，以簡化您管理風險的方式，以及遵守法規和業界標準的方式。

韌性 AWS CodeDeploy

AWS 全球基礎架構是圍繞區 AWS 域和可用區域建立的。 AWS 區域提供多個實體分離和隔離的可用區域，這些區域透過低延遲、高輸送量和高度備援的網路連線。透過可用區域，您可以設計與操作的應用程式和資料庫，在可用區域之間自動容錯移轉而不會發生中斷。可用區域的可用性、容錯能力和擴展能力，均較單一或多個資料中心的傳統基礎設施還高。

如需區域和可用區域的詳 AWS 細資訊，請參閱AWS 全域基礎結構。

基礎結構安全 AWS CodeDeploy

身為受管服務， AWS CodeDeploy 受 Amazon Web Services：安 AWS 全程序概觀白皮書中所述的全球網路安全程序保護。

您可以使用 AWS 已發佈的 API 呼叫透 CodeDeploy 過網路進行存取。用戶端必須支援 Transport Layer Security (TLS) 1.2 或更新版本。建議使用 TLS 1.3 或更新版本。用戶端也必須支援具備完美轉送私密 (PFS) 的密碼套件，例如臨時 Diffie-Hellman (DHE) 或橢圓曲線臨時 Diffie-Hellman (ECDHE)。現代系統(如 Java 7 和更新版本)大多會支援這些模式。

請求必須使用存取金鑰 ID 和與 IAM 委託人相關聯的私密存取金鑰來簽署。或者，您可以使用 AWS Security Token Service (AWS STS) 以產生暫時安全憑證以簽署請求。

參考資料

參考。

主題

- CodeDeploy AppSpec 檔案參考
- CodeDeploy 用戶端組態參考
- AWS CloudFormation CodeDeploy供參考的範本
- 搭 CodeDeploy 配 Amazon Virtual Private Cloud 使用
- CodeDeploy 資源套件參考
- CodeDeploy 配額

CodeDeploy AppSpec 檔案參考

本節僅供參考。如需 AppSpec 檔案的概念性概述，請參閱〈〉Application Specification Files。

應用程式規格檔AppSpec 案 (檔案) 是用 CodeDeploy 來管理部署的 YAML 格式或 JSON 格式檔案。

> **Note**
>
> 除非您正在執行本機部署`appspec.yml`，否則必須命名 EC2/ 內部部署的 AppSpec 檔案。如需詳細資訊，請參閱 建立本機部署。

主題

- AppSpec Amazon ECS 運算平台上的檔案
- AppSpec AWS Lambda 運算平台上的檔案
- AppSpec EC2/內部部署計算平台上的檔案
- AppSpec 檔案結構
- AppSpec 檔案範例
- AppSpec 檔案間距
- 驗證您的 AppSpec 檔案和檔案位置

AppSpec Amazon ECS 運算平台上的檔案

對於 Amazon ECS 運算平台應用程式，會使用 AppSpec 檔案 CodeDeploy 來判斷：

- 您的 Amazon ECS 任務定義檔案。這在 AppSpec 文件中的指`TaskDefinition`令中以其 ARN 指定。
- 取代工作集中的容器和連接埠，應用程式負載平衡器或 Network Load Balancer 會在部署期間重新路由傳送流量。這是通過 AppSpec 文件中的指`LoadBalancerInfo`令指定的。
- 有關 Amazon ECS 服務的選用資訊，例如執行所在的平台版本、子網路及其安全群組。
- 可在 Amazon ECS 部署期間與生命週期事件相對應的掛接期間執行的選用 Lambda 函數。如需詳細資訊，請參閱 AppSpec Amazon ECS 部署的「掛鉤」部分。

AppSpec AWS Lambda 運算平台上的檔案

對於 AWS Lambda 運算平台應用程式，會使用該 AppSpec 檔案 CodeDeploy 來判斷：

- 要部署哪個 Lambda 函數版本。
- 哪些 Lambda 函數用作驗證測試。

AppSpec 檔案可以是 YAML 格式或 JSON 格式。您也可以在建立部署時，將 AppSpec 檔案內容直接輸入 CodeDeploy 主控台。

AppSpec EC2/內部部署計算平台上的檔案

如果您的應用程式使用 EC2 /內部部署計算平台，該 AppSpec 檔案必須是名為 YAML 格式的檔案，`appspec.yml`而且必須放置在應用程式原始程式碼目錄結構的根目錄結構中。否則，部署會失敗。它被用 CodeDeploy 來確定：

- 它應該從 Amazon S3 或 GitHub. 中的應用程式修訂版安裝到執行個體上的項目。
- 為回應部署生命週期事件而執行的生命週期事件勾點。

在您有完成的 AppSpec 檔案之後，您可以將它與要部署的內容一起捆綁到歸檔檔案 (zip、tar 或壓縮的 tar) 中。如需詳細資訊，請參閱 使用的應用程式修訂 CodeDeploy。

> **Note**
>
> 視窗伺服器執行個體不支援 tar 和壓縮的 tar 封存檔案格式 (.tar 和 .tar.gz)。

在您擁有隨附的封存檔案 (在修訂版中稱 CodeDeploy 為) 之後，您可以將其上傳到 Amazon S3 儲存貯體或 Git 儲存庫。然後，您可 CodeDeploy 以使用部署修訂版本。如需說明，請參閱使用建立部署 CodeDeploy。

EC2 /內部部署計算平台部署的 appspec.yml 會儲存在修訂版本的根目錄中。如需詳細資訊，請參閱 新增EC2/ AppSpec 內部部署的檔案 及 規劃修訂 CodeDeploy。

AppSpec 檔案結構

以下是用於部署至 AWS Lambda 和 EC2 /內部部署計算平台的檔 AppSpec 案的高階結構。

除非另有指定，否則 YAML 格式 AppSpec 檔案中為字串的值不得以引號 (「」) 括住。

AppSpec Amazon ECS 部署的檔案結構

> **Note**
>
> 這個 AppSpec 文件是用 YAML 編寫的，但是您可以使用相同的結構在 JSON 中寫入一個文件。JSON 格式 AppSpec 檔案中的字串一律會以引號 (「」) 括住。

```
version: 0.0
resources:
  ecs-service-specifications
hooks:
  deployment-lifecycle-event-mappings
```

在此結構中：

version

本節指定 AppSpec 檔案的版本。請不要變更此值。這是必要的。目前，唯一允許的值為 **0.0**。它被保留以 CodeDeploy 備 future 使用。

以字串指定 version。

resources

本節指定要部署之 Amazon ECS 應用程式的相關資訊。

如需詳細資訊，請參閱 AppSpec Amazon ECS 部署的「資源」部分。

hooks

本節指定要在特定部署生命週期事件掛接上執行的 Lambda 函數，以驗證部署。

如需詳細資訊，請參閱 Amazon ECS 部署的生命週期事件掛鉤清單。

AppSpec AWS Lambda 部署的檔案結構

> **Note**
>
> 這個 AppSpec 檔案是以 YAML 撰寫的，但是您可以使用相同的結構，為 JSON 中的 Lambda 部署撰寫 AppSpec 檔案。JSON 格式 AppSpec 檔案中的字串一律會以引號 (「」) 括住。

```
version: 0.0
resources:
  lambda-function-specifications
hooks:
  deployment-lifecycle-event-mappings
```

在此結構中：

version

本節指定 AppSpec 檔案的版本。請不要變更此值。這是必要的。目前，唯一允許的值為 **0.0**。它被保留以 CodeDeploy 備 future 使用。

以字串指定 version。

resources

本節指定要部署之 Lambda 函數的相關資訊。

如需詳細資訊，請參閱 AppSpec 「資源」區段 (僅限 Amazon ECS 和 AWS Lambda 部署)。

hooks

本節指定要在特定部署生命週期事件執行的 Lambda 函數，以驗證部署。

如需詳細資訊，請參閱 AppSpec 「掛鉤」部分。

AppSpec EC2/內部部署的檔案結構

```
version: 0.0
os: operating-system-name
files:
  source-destination-files-mappings
permissions:
  permissions-specifications
hooks:
  deployment-lifecycle-event-mappings
```

在此結構中：

version

本節指定 AppSpec 檔案的版本。請不要變更此值。這是必要的。目前，唯一允許的值為 **0.0**。它被保留以 CodeDeploy 備 future 使用。

以字串指定 version。

os

本區段指定您要部署之執行個體的作業系統值。這是必要的。您可以指定的值如下：

- Linux — 執行個體是 Amazon Linux、Ubuntu 伺服器或 RHEL 執行個體。
- 視窗 — 執行個體是 Windows 伺服器執行個體。

以字串指定 os。

files

本區段指定在部署 Install 事件期間應該複製至執行個體的檔案名稱。

如需詳細資訊，請參閱 AppSpec 「檔案」區段 (僅適用於 EC2 /內部部署)。

permissions

本區段指定應該如何將特殊許可 (如果有的話) 套用至 `files` 區段中的檔案，因為它們會複製至執行個體。本節僅適用於 Amazon Linux、Ubuntu 伺服器和 RHEL (RHEL) 執行個體。

若要取得更多資訊，請參閱AppSpec 「權限」區段 (僅適用於 EC2 /內部部署)。

hooks

本區段指定要在部署期間於特定部署生命週期事件執行的指令碼。

如需詳細資訊，請參閱 AppSpec 「掛鉤」部分。

主題

- AppSpec 「檔案」區段 (僅適用於 EC2 /內部部署)
- AppSpec 「資源」區段 (僅限 Amazon ECS 和 AWS Lambda 部署)
- AppSpec 「權限」區段 (僅適用於 EC2 /內部部署)
- AppSpec 「掛鉤」部分

AppSpec 「檔案」區段 (僅適用於 EC2 /內部部署)

提供 CodeDeploy 有關在部署的 Install 事件期間，應在執行個體上安裝應該從應用程式修訂版本中的哪些檔案的相關資訊。唯有您在部署期間將修訂中的檔案複製至執行個體上的位置時，會需要本區段。

本區段的結構如下：

```
files:
  - source: source-file-location-1
    destination: destination-file-location-1
file_exists_behavior: DISALLOW|OVERWRITE|RETAIN
```

您可以設定多個 source 和 destination 配對。

source 說明識別要從修訂複製至執行個體的檔案或目錄：

- 如果 source 是指檔案，只會將指定的檔案複製至執行個體。
- 如果 source 是指目錄，會將該目錄中的所有檔案都複製至執行個體。
- 如果source是單一斜線 (Amazon Linux、RHEL 和 Ubuntu 伺服器執行個體為「/」，或是 Windows 伺服器執行個體的「\」)，則修訂版本中的所有檔案都會複製到執行個體。

中使用的路徑source是相對於appspec.yml檔案的路徑，該路徑應位於修訂的根目錄。如需修訂檔案結構的詳細資訊，請參閱規劃修訂 CodeDeploy。

destination 說明識別執行個體上應該複製檔案的位置。這必須是一個完全合格的路徑，例如/root/destination/directory（在 Linux，RHEL 和 Ubuntu 上）或c:\destination\folder（在視窗上）。

source 和 destination 各以字串指定。

此指file_exists_behavior令是選擇性的，並指定如何 CodeDeploy處理已存在於部署目標位置中，但不是先前成功部署的一部分的檔案。此設定可採用下列任一值：

- 不允許：部署失敗。如果未指定選項，這也是預設行為。
- 覆寫：目前部署的應用程式修訂版本中的檔案版本會取代執行個體上已有的版本。
- RETAIN：系統會保留執行個體上已存在的檔案版本，並做為新部署的一部分使用。

使用此file_exists_behavior設定時，請瞭解此設定：

- 只能指定一次，並套用至下列出的所有檔案和目錄files:。
- 優先於選--file-exists-behavior AWS CLI 項和 fileExistsBehavior API 選項（兩者都是可選的）。

以下是 Amazon Linux、Ubuntu 伺服器或 RHEL 執行個體的範例files區段。

```
files:
  - source: Config/config.txt
    destination: /webapps/Config
  - source: source
    destination: /webapps/myApp
```

在本範例中，會在 Install 事件期間執行下列兩個操作：

1. 將修訂中的 Config/config.txt 檔案複製至執行個體上的 /webapps/Config/config.txt 路徑。
2. 將修訂 source 目錄中的所有檔案都遞迴複製至執行個體上的 /webapps/myApp 目錄。

「文件」部分的例子

下列範例顯示如何指定 files 區段。儘管這些示例描述了 Windows 服務器文件和目錄（文件夾）結構，但它們可以很容易地適用於 Amazon Linux，Ubuntu 服務器和 RHEL 實例。

> **Note**
>
> 只有 EC2 /內部部署使用此區段。`files`它不適用於 AWS Lambda 部署。

在下列範例中，假設這些檔案出現在 `source` 根目錄的套件中：

- `appspec.yml`
- `my-file.txt`
- `my-file-2.txt`
- `my-file-3.txt`

```
# 1) Copy only my-file.txt to the destination folder c:\temp.
#
files:
  - source: .\my-file.txt
    destination: c:\temp
#
# Result:
#   c:\temp\my-file.txt
#
# ----------------------
#
# 2) Copy only my-file-2.txt and my-file-3.txt to the destination folder c:\temp.
#
files:
  - source: my-file-2.txt
    destination: c:\temp
  - source: my-file-3.txt
    destination: c:\temp
#
# Result:
#   c:\temp\my-file-2.txt
#   c:\temp\my-file-3.txt
#
# ----------------------
#
# 3) Copy my-file.txt, my-file-2.txt, and my-file-3.txt (along with the appspec.yml
 file) to the destination folder c:\temp.
#
```

```
files:
  - source: \
    destination: c:\temp
#
# Result:
#   c:\temp\appspec.yml
#   c:\temp\my-file.txt
#   c:\temp\my-file-2.txt
#   c:\temp\my-file-3.txt
```

在下列範例中，假設 appspec.yml 出現在 source 根目錄以及名為 my-folder 資料夾 (包含三個檔案) 的套件中：

- appspec.yml
- my-folder\my-file.txt
- my-folder\my-file-2.txt
- my-folder\my-file-3.txt

```
# 4) Copy the 3 files in my-folder (but do not copy my-folder itself) to the
 destination folder c:\temp.
#
files:
  - source: .\my-folder
    destination: c:\temp
#
# Result:
#   c:\temp\my-file.txt
#   c:\temp\my-file-2.txt
#   c:\temp\my-file-3.txt
#
# ---------------------
#
# 5) Copy my-folder and its 3 files to my-folder within the destination folder c:\temp.
#
files:
  - source: .\my-folder
    destination: c:\temp\my-folder
#
# Result:
#   c:\temp\my-folder\my-file.txt
#   c:\temp\my-folder\my-file-2.txt
```

```
#   c:\temp\my-folder\my-file-3.txt
#
# ---------------------
#
# 6) Copy the 3 files in my-folder to other-folder within the destination folder c:
\temp.
#
files:
  - source: .\my-folder
    destination: c:\temp\other-folder
#
# Result:
#   c:\temp\other-folder\my-file.txt
#   c:\temp\other-folder\my-file-2.txt
#   c:\temp\other-folder\my-file-3.txt
#
# ---------------------
#
# 7) Copy only my-file-2.txt and my-file-3.txt to my-folder within the destination
 folder c:\temp.
#
files:
  - source: .\my-folder\my-file-2.txt
    destination: c:\temp\my-folder
  - source: .\my-folder\my-file-3.txt
    destination: c:\temp\my-folder
#
# Result:
#   c:\temp\my-folder\my-file-2.txt
#   c:\temp\my-folder\my-file-3.txt
#
# ---------------------
#
# 8) Copy only my-file-2.txt and my-file-3.txt to other-folder within the destination
 folder c:\temp.
#
files:
  - source: .\my-folder\my-file-2.txt
    destination: c:\temp\other-folder
  - source: .\my-folder\my-file-3.txt
    destination: c:\temp\other-folder
#
# Result:
#   c:\temp\other-folder\my-file-2.txt
```

```
#   c:\temp\other-folder\my-file-3.txt
#
# ---------------------
#
# 9) Copy my-folder and its 3 files (along with the appspec.yml file) to the
 destination folder c:\temp. If any of the files already exist on the instance,
 overwrite them.
#
files:
  - source: \
    destination: c:\temp
file_exists_behavior: OVERWRITE
#
# Result:
#   c:\temp\appspec.yml
#   c:\temp\my-folder\my-file.txt
#   c:\temp\my-folder\my-file-2.txt
#   c:\temp\my-folder\my-file-3.txt
```

AppSpec 「資源」區段 (僅限 Amazon ECS 和 AWS Lambda 部署)

根據部署的運算平台，AppSpec 檔案`'resources'`區段中的內容會有所不同。Amazon ECS 部署的部`'resources'`分包含您的 Amazon ECS 任務定義、用於將流量路由到更新後的 Amazon ECS 任務集的容器和連接埠，以及其他選用資訊。 AWS Lambda 部署的`'resources'`區段包含 Lambda 函數的名稱、別名、目前版本和目標版本。

主題

- AppSpec AWS Lambda 部署的「資源」部分
- AppSpec Amazon ECS 部署的「資源」部分

AppSpec AWS Lambda 部署的「資源」部分

此`'resources'`區段會指定要部署的 Lambda 函數，並具有下列結構：

YAML：

```
resources:
  - name-of-function-to-deploy:
      type: "AWS::Lambda::Function"
      properties:
        name: name-of-lambda-function-to-deploy
```

```
        alias: alias-of-lambda-function-to-deploy
        currentversion: version-of-the-lambda-function-traffic-currently-points-to
        targetversion: version-of-the-lambda-function-to-shift-traffic-to
```

JSON：

```
"resources": [
    {
        "name-of-function-to-deploy" {
            "type": "AWS::Lambda::Function",
            "properties": {
                "name": "name-of-lambda-function-to-deploy",
                "alias": "alias-of-lambda-function-to-deploy",
                "currentversion": "version-of-the-lambda-function-traffic-currently-
points-to",
                "targetversion": "version-of-the-lambda-function-to-shift-traffic-to"
            }
        }
    }
]
```

每個屬性皆以字串指定。

- `name` - 必要。這是要部署的 Lambda 函數的名稱。
- `alias` - 必要。這是 Lambda 函數的別名名稱。
- `currentversion` - 必要。這是 Lambda 函數流量目前指向的版本。此值必須是有效的正整數。
- `targetversion` - 必要。這是 Lambda 函數流量轉移到的版本。此值必須是有效的正整數。

AppSpec Amazon ECS 部署的「資源」部分

本`'resources'`節指定要部署的 Amazon ECS 服務，其結構如下：

YAML：

```
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "task-definition-arn"
        LoadBalancerInfo:
          ContainerName: "ecs-container-name"
```

```
          ContainerPort: "ecs-application-port"
# Optional properties
        PlatformVersion: "ecs-service-platform-version"
        NetworkConfiguration:
          AwsvpcConfiguration:
            Subnets: ["ecs-subnet-1","ecs-subnet-n"]
            SecurityGroups: ["ecs-security-group-1","ecs-security-group-n"]
            AssignPublicIp: "ENABLED | DISABLED"
        CapacityProviderStrategy:
          - Base: integer
            CapacityProvider: "capacityProviderA"
            Weight: integer
          - Base: integer
            CapacityProvider: "capacityProviderB"
            Weight: integer
```

JSON：

```
"Resources": [
    {
        "TargetService": {
            "Type": "AWS::ECS::Service",
            "Properties": {
                "TaskDefinition": "task-definition-arn",
                "LoadBalancerInfo": {
                    "ContainerName": "ecs-container-name",
                    "ContainerPort": "ecs-application-port"
                },
                "PlatformVersion": "ecs-service-platform-version",
                "NetworkConfiguration": {
                    "AwsvpcConfiguration": {
                        "Subnets": [
                            "ecs-subnet-1",
                            "ecs-subnet-n"
                        ],
                        "SecurityGroups": [
                            "ecs-security-group-1",
                            "ecs-security-group-n"
                        ],
                        "AssignPublicIp": "ENABLED | DISABLED"
                    }
                },
                "CapacityProviderStrategy": [
```

```
                    {
                        "Base": integer,
                        "CapacityProvider": "capacityProviderA",
                        "Weight": integer
                    },
                    {
                        "Base": integer,
                        "CapacityProvider": "capacityProviderB",
                        "Weight": integer
                    }
                ]
            }
        }
    }
]
```

每個屬性都使用一個字串 (除了ContainerPort:數字) 來指定。

- TaskDefinition - 必要。這是要部署之 Amazon ECS 服務的任務定義。這包含任務定義的 ARN 指定。ARN 的格式為 arn:aws:ecs:*aws-region*:*account-id*:task-definition/*task-definition-family*:*task-definition-revision*。如需詳細資訊，請參閱 Amazon 資源名稱 (ARN) 和 AWS 服務命名空間。

  > **Note**
  >
  > ARN 的:*task-definition-revision*部分是可選的。如果省略它，Amazon ECS 會使用任務定義的最新作用中修訂版本。

- ContainerName - 必要。這是包含您的 Amazon ECS 應用程式的 Amazon ECS 容器的名稱。它必須是 Amazon ECS 任務定義中指定的容器。
- ContainerPort - 必要。這是要將流量路由到的容器上的連接埠。
- PlatformVersion：選用。已部署的 Amazon ECS 服務中 Fargate 任務的平台版本。如需詳細資訊，請參閱 AWS Fargate 平台版本。如果未指定，LATEST則依預設使用。
- NetworkConfiguration：選用。在 AwsvpcConfiguration 下，您可以指定下列項目。如需詳細資訊，請參閱 Amazon ECS 容器服務 API 參考AwsVpcConfiguration中的。
  - Subnets：選用。Amazon ECS 服務中一或多個子網路的逗號分隔清單。
  - SecurityGroups：選用。Amazon 彈性容器服務中一或多個安全群組的逗號分隔清單。

- `AssignPublicIp`：選用。字串；指定 Amazon ECS 服務的 elastic network interface 是否接收公有 IP 位址。有效值為 `ENABLED` 和 `DISABLED`。

> **Note**
>
> `NetworkConfiguration` 下的設定必須全部指定，或都不指定。例如，如果您想要指定 `Subnets`，您還必須指定 `SecurityGroups` 和 `AssignPublicIp`。如果未指定任何項目，則 CodeDeploy 會使用目前的網路 Amazon ECS 設定。

- `CapacityProviderStrategy`：選用。您要用於部署的 Amazon ECS 容量提供者清單。如需詳細資訊，請參閱 Amazon 彈性容器服務開發人員指南中的 Amazon ECS 容量提供者。您可以為每個容量提供者指定下列設定。有關這些設定的詳細資訊，請參閱《AWS CloudFormation 使用指南》AWS::ECS::ServiceCapacityProviderStrategyItem中的
  - `Base`：選用。基礎值指定至少要在指定容量提供者上執行多少任務數量。容量提供者策略中只有一個容量提供者可以定義基礎。如果未指定任何值，則會使用預設值 0。
  - `CapacityProvider`：選用。容量提供者的簡短名稱。示例：capacityProviderA 提供
  - `Weight`：選用。

    加權值會指定應使用指定容量提供者之已啟動任務總數的相對百分比。如果已定義，則會在滿足 `base` 值之後考量 `weight` 值。

    如果未指定任何 `weight` 值，則會使用 `0` 的預設值。在容量提供者策略中指定多個容量供應商時，至少有一個容量供應商必須具有大於零的加權值，且不會使用具有加權 `0` 的任何容量提供者來放置任務。如果您在策略中指定多個容量提供者均具有加權 `0`，則使用容量提供者策略的任何 `RunTask` 或 `CreateService` 動作都會失敗。

    使用加權的一個範例案例為定義策略，該策略包含兩個容量提供者，而且兩者都具有 1 加權，則當滿足 `base` 時，任務將會平均分割到兩個容量提供者。依此邏輯，如果您為 capacityProviderA 指定 1 的權重，並為 capacityProviderB 指定 4 的權重，則對於使用 capacityProviderA 執行的每一個任務而言，四個任務將使用 capacityProviderB。

AppSpec 「權限」區段 (僅適用於 EC2 /內部部署)

`'permissions'` 區段指定在將 `'files'` 區段中的檔案和目錄/資料夾複製至執行個體之後，應如何將特殊許可 (如果有的話) 套用至檔案以及目錄/資料夾。您可以指定多個 `object` 說明。此區段為選用。它僅適用於 Amazon Linux、Ubuntu 伺服器和 RHEL 執行個體。

> **Note**
>
> 此'permissions'區段僅適用於 EC2 /內部部署。它不用於 AWS Lambda 或 Amazon ECS 部署。

本區段的結構如下：

```
permissions:
  - object: object-specification
    pattern: pattern-specification
    except: exception-specification
    owner: owner-account-name
    group: group-name
    mode: mode-specification
    acls:
      - acls-specification
    context:
      user: user-specification
      type: type-specification
      range: range-specification
    type:
      - object-type
```

說明如下：

- object - 必要。將檔案系統物件複製至執行個體之後，將套用指定之許可的一組檔案系統物件 (檔案或目錄/資料夾)。

  以字串指定 object。

- pattern - 選用。指定要套用許可的模式。如果未指定或已指定特殊字元 "**"，會根據 type 將許可套用至所有相符的檔案或目錄。

  使用含引數 ("") 的字串指定 pattern。

- except - 選用。指定 pattern 例外狀況的任何檔案或目錄。

  以方括號內逗號分隔的字串清單，指定 except。

- owner - 選用。object 的擁有者名稱。如果未指定，在複製操作之後，所有套用至原始檔案或目錄/資料夾結構的現有擁有者都不會受到影響。

以字串指定 owner。

- group - 選用。object 的群組名稱。如果未指定，在複製操作之後，所有套用至原始檔案或目錄/資料夾結構的現有群組都不會受到影響。

  以字串指定 group。

- mode - 選用。指定要套用之權限的數值object。該模式設置遵循 Linux chmod 命令語法。

  > **Important**
  >
  > 如果該值包括前導零，則必須用雙引號括住它，或者刪除前導零，以便僅保留三位數。

  > **Note**
  >
  > mode設定不支援符號表示法 (例如)。**u+x**

  範例：

  - mode: "0644"授予物件擁有者的讀取和寫入權限 (6)、群組的唯讀權限 (4)，以及所有其他使用者的唯讀權限 (4)。
  - mode: 644授予與相同的權限mode: "0644"。
  - mode: 4755設置 setuid 屬性（4），賦予所有者完全控制權限（7），授予該組的讀取和執行權限（5），並授予所有其他用戶讀取和執行權限（5）。

    如需更多範例，請參閱 Linux chmod 指令文件。

    如果未指定 mode，則在複製操作之後，套用至原始檔案或資料夾結構的所有現有模式都會保持不變。

- acls - 選用。字元字串清單，代表套用至 object 的一或多個存取控制清單 (ACL) 項目。例如，**u:bob:rw** 代表 **bob** 使用者的讀取和寫入許可 (如需範例，請參閱 Linux setfacl 命令文件中的 ACL 項目格式範例)。您可以指定多個 ACL 項目。如果未指定 acls，在複製操作之後，任何套用至原始檔案或目錄/資料夾結構的現有 ACL 都不會受到影響。這些會取代現有的所有 ACL。

  使用依序後接一個空格和一個字串的破折號 (-) 指定 acls (例如，- u:jane:rw)。如果您有多個 ACL，會在個別的行指定。

> **Note**
>
> 設定未命名的使用者、未命名群組或其他類似 ACL 項目會導致 AppSpec 檔案失敗。使用 `mode` 指定這些類型的許可。

- `context` - 選用。對於 Security-Enhanced Linux (SELinux) 啟動的執行個體，是套用到複製物件的安全相關內容標籤清單。標籤指定為包含 `user`、`type` 和 `range` 的索引鍵。(如需詳細資訊，請參閱 SELinux 文件)。每個金鑰都是以字串形式輸入。如果未指定，則在複製作業之後，任何套用至原始檔案或目錄/資料夾結構的現有標籤都不受影響。
  - `user` - 選用。SELinux 使用者。
  - `type` - 選用。SELinux 類型名稱。
  - `range` - 選用。SELinux 範圍的指標。這不會有任何影響，除非機器上有啟用 Multi-Level Security (MLS) 和 Multi-Category Security (MCS)。如果未啟用，`range` 預設為 **`s0`**。

  以字串指定 `context` (例如，`user: unconfined_u`)。每個 `context` 指定於不同行。
- `type` - 選用。要套用指定權限的物件類型。`type` 是字串，可設定為 **`file`** 或 **`directory`**。如果指定 **`file`**，許可只會套用到立即包含在複製操作後的 `object` (而非 `object` 本身) 中的檔案。如果指定 **`directory`**，使用權限會以遞迴方式套用到位於複製操作後 `object` (但不是 `object` 本身) 中任何地方的所有目錄/資料夾。

  以破折號 (-) 後接一個空格和一個字串的形式指定 `type` (例如，`- file`)。

「權限」部分示例

以下範例顯示如何使用 `object`、`pattern`、`except`、`owner`、`mode` 和 `type` 說明指定 `'permissions'` 區段。此範例僅適用於 Amazon Linux、Ubuntu 伺服器和 RHEL 執行個體。在這個範例中，假設以下檔案和資料夾都複製到下列階層中的執行個體：

```
/tmp
  `-- my-app
       |-- my-file-1.txt
       |-- my-file-2.txt
       |-- my-file-3.txt
       |-- my-folder-1
       |     |-- my-file-4.txt
       |     |-- my-file-5.txt
       |     `-- my-file-6.txt
       `-- my-folder-2
```

```
             |-- my-file-7.txt
             |-- my-file-8.txt
             |-- my-file-9.txt
            `-- my-folder-3
```

下列 AppSpec 檔案示範如何在複製這些檔案和資料夾之後設定權限：

```
version: 0.0
os: linux
# Copy over all of the folders and files with the permissions they
#  were originally assigned.
files:
  - source: ./my-file-1.txt
    destination: /tmp/my-app
  - source: ./my-file-2.txt
    destination: /tmp/my-app
  - source: ./my-file-3.txt
    destination: /tmp/my-app
  - source: ./my-folder-1
    destination: /tmp/my-app/my-folder-1
  - source: ./my-folder-2
    destination: /tmp/my-app/my-folder-2
# 1) For all of the files in the /tmp/my-app folder ending in -3.txt
#  (for example, just my-file-3.txt), owner = adm, group = wheel, and
#  mode = 464 (-r--rw-r--).
permissions:
  - object: /tmp/my-app
    pattern: "*-3.txt"
    owner: adm
    group: wheel
    mode: 464
    type:
      - file
# 2) For all of the files ending in .txt in the /tmp/my-app
#  folder, but not for the file my-file-3.txt (for example,
#  just my-file-1.txt and my-file-2.txt),
#  owner = ec2-user and mode = 444 (-r--r--r--).
  - object: /tmp/my-app
    pattern: "*.txt"
    except: [my-file-3.txt]
    owner: ec2-user
    mode: 444
    type:
```

```
      - file
# 3) For all the files in the /tmp/my-app/my-folder-1 folder except
#  for my-file-4.txt and my-file-5.txt, (for example,
#  just my-file-6.txt), owner = operator and mode = 646 (-rw-r--rw-).
  - object: /tmp/my-app/my-folder-1
    pattern: "**"
    except: [my-file-4.txt, my-file-5.txt]
    owner: operator
    mode: 646
    type:
      - file
# 4) For all of the files that are immediately under
#  the /tmp/my-app/my-folder-2 folder except for my-file-8.txt,
#  (for example, just my-file-7.txt and
#  my-file-9.txt), owner = ec2-user and mode = 777 (-rwxrwxrwx).
  - object: /tmp/my-app/my-folder-2
    pattern: "**"
    except: [my-file-8.txt]
    owner: ec2-user
    mode: 777
    type:
      - file
# 5) For all folders at any level under /tmp/my-app that contain
#  the name my-folder but not
#  /tmp/my-app/my-folder-2/my-folder-3 (for example, just
#  /tmp/my-app/my-folder-1 and /tmp/my-app/my-folder-2),
#  owner = ec2-user and mode = 555 (dr-xr-xr-x).
  - object: /tmp/my-app
    pattern: "*my-folder*"
    except: [tmp/my-app/my-folder-2/my-folder-3]
    owner: ec2-user
    mode: 555
    type:
      - directory
# 6) For the folder /tmp/my-app/my-folder-2/my-folder-3,
#  group = wheel and mode = 564 (dr-xrw-r--).
  - object: /tmp/my-app/my-folder-2/my-folder-3
    group: wheel
    mode: 564
    type:
      - directory
```

產生的許可，如下所示：

```
-r--r--r-- ec2-user root  my-file-1.txt
-r--r--r-- ec2-user root  my-file-2.txt
-r--rw-r-- adm      wheel my-file-3.txt

dr-xr-xr-x ec2-user root  my-folder-1
-rw-r--r-- root     root  my-file-4.txt
-rw-r--r-- root     root  my-file-5.txt
-rw-r--rw- operator root  my-file-6.txt

dr-xr-xr-x ec2-user root  my-folder-2
-rwxrwxrwx ec2-user root  my-file-7.txt
-rw-r--r-- root     root  my-file-8.txt
-rwxrwxrwx ec2-user root  my-file-9.txt

dr-xrw-r-- root     wheel my-folder-3
```

以下範例說明如何指定 'permissions' 區段，並新增 acls 和 context 說明。此範例僅適用於 Amazon Linux、Ubuntu 伺服器和 RHEL 執行個體。

```
permissions:
  - object: /var/www/html/WordPress
    pattern: "**"
    except: [/var/www/html/WordPress/ReadMe.txt]
    owner: bob
    group: writers
    mode: 644
    acls:
      - u:mary:rw
      - u:sam:rw
      - m::rw
    context:
      user: unconfined_u
      type: httpd_sys_content_t
      range: s0
    type:
      - file
```

AppSpec 「掛鉤」部分

根據您部署的運算平台， AppSpec 檔案'hooks'區段中的內容會有所不同。EC2 /內部部署的區'hooks'段包含將部署生命週期事件掛接連結至一或多個指令碼的對應。Lambda 或 Amazon ECS 部署的'hooks'區段會指定要在部署生命週期事件期間執行的 Lambda 驗證函數。如果事件勾點不存

在，則不會為該事件執行任何操作。只有在執行指令碼或 Lambda 驗證函數做為部署的一部分時，才需要本節。

主題

- AppSpec Amazon ECS 部署的「掛鉤」部分
- AppSpec AWS Lambda 部署的「掛鉤」部分
- AppSpec EC2 /內部部署的「掛鉤」部分

AppSpec Amazon ECS 部署的「掛鉤」部分

主題

- Amazon ECS 部署的生命週期事件掛鉤清單
- 在 Amazon ECS 部署中執行掛鉤的順序。
- 「掛鉤」部分的結構
- 示例 Lambda「掛鉤」功能

Amazon ECS 部署的生命週期事件掛鉤清單

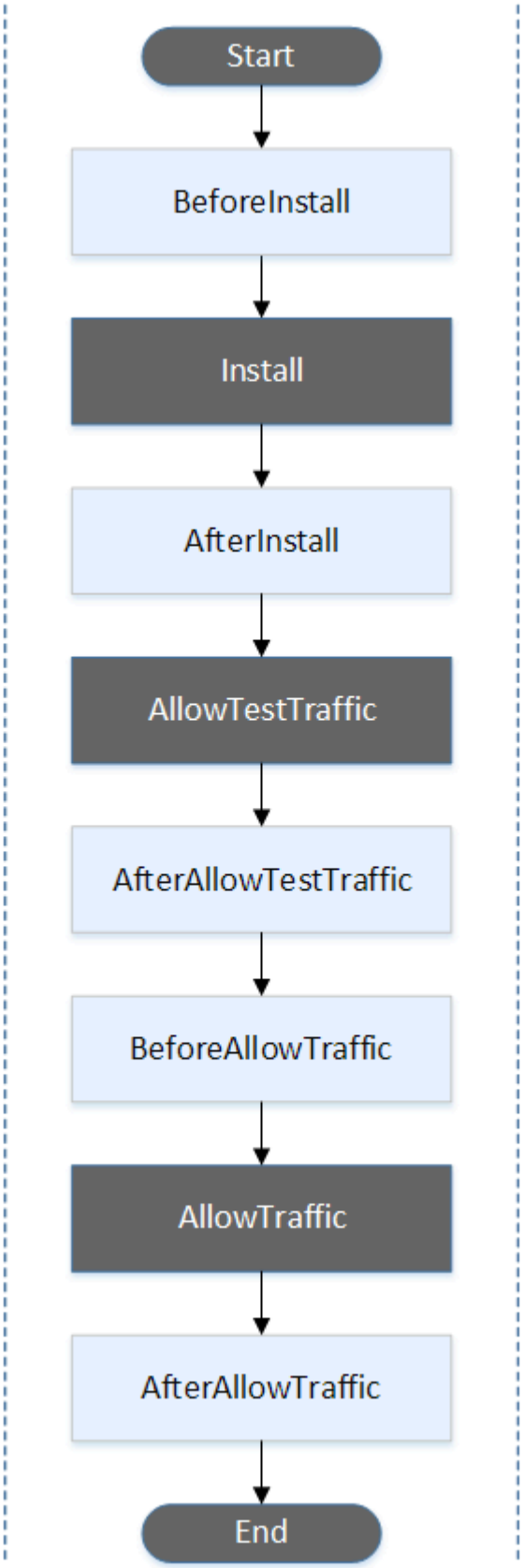
AWS Lambda 掛接是一個 Lambda 函數，在生命週期事件名稱後面的新行上以字串指定。每個部署執行每個勾點一次。以下是您可以在 Amazon ECS 部署期間執行勾點的生命週期事件說明。

- `BeforeInstall`— 用於在建立取代任務集之前執行工作。單一目標群組與原始任務設定相關。如果指定了選用測試接聽程式，則和原始任務設定相關。目前無法轉返。
- `AfterInstall`— 用於在建立取代任務集且其中一個目標群組與其關聯之後執行工作。如果指定了選用測試接聽程式，則和原始任務設定相關。此生命週期事件上的勾點函數結果可以觸發轉返。
- `AfterAllowTestTraffic`— 用於在測試接聽程式將流量提供給取代工作集之後執行工作。目前的勾點函數結果可以觸發轉返。
- `BeforeAllowTraffic`— 用於在第二個目標群組與取代任務集相關聯之後，但在流量轉移至取代任務集之前執行工作。此生命週期事件上的勾點函數結果可以觸發轉返。
- `AfterAllowTraffic`— 用於在第二個目標群組為取代工作集提供流量之後執行工作。此生命週期事件上的勾點函數結果可以觸發轉返。

如需詳細資訊，請參閱 Amazon ECS 部署期間會發生什麼情況 及 教學課程：透過驗證測試部署 Amazon ECS 服務。

在 Amazon ECS 部署中執行掛鉤的順序。

在 Amazon ECS 部署中，事件掛接會以下列順序執行：


Start
BeforeInstall
Install
AfterInstall
AllowTestTraffic
AfterAllowTestTraffic
BeforeAllowTraffic
AllowTraffic
AfterAllowTraffic
End

> **Note**
>
> 部署中的「開始」TestTraffic、「安裝」AllowTraffic、和「結束」事件無法編寫指令碼，這就是為什麼它們在此圖表中以灰色顯示的原因。

「掛鉤」部分的結構

下列為 'hooks' 部分的結構範例。

使用 YAML：

```
Hooks:
  - BeforeInstall: "BeforeInstallHookFunctionName"
  - AfterInstall: "AfterInstallHookFunctionName"
  - AfterAllowTestTraffic: "AfterAllowTestTrafficHookFunctionName"
  - BeforeAllowTraffic: "BeforeAllowTrafficHookFunctionName"
  - AfterAllowTraffic: "AfterAllowTrafficHookFunctionName"
```

使用 JSON：

```
"Hooks": [
  {
   "BeforeInstall": "BeforeInstallHookFunctionName"
  },
  {
   "AfterInstall": "AfterInstallHookFunctionName"
  },
  {
   "AfterAllowTestTraffic": "AfterAllowTestTrafficHookFunctionName"
  },
  {
   "BeforeAllowTraffic": "BeforeAllowTrafficHookFunctionName"
  },
  {
   "AfterAllowTraffic": "AfterAllowTrafficHookFunctionName"
  }
 ]
}
```

示例 Lambda「掛鉤」功能

使用此'hooks'區段指定 CodeDeploy 可呼叫以驗證 Amazon ECS 部署的 Lambda 函數。您可以針對BeforeInstall、AfterInstall、和AfterAllowTraffic部署生命週期事件使用相同的函數或不同的函數。AfterAllowTestTraffic BeforeAllowTraffic完成驗證測試後，Lambda AfterAllowTraffic 函數會回呼 CodeDeploy並提供Succeeded或的結果Failed。

> **Important**
>
> 如果 CodeDeploy Lambda 驗證函數在一小時內未收到通知，則會將部署視為已失敗。

在叫用 Lambda 勾點函數之前，必須使用命令通知伺服器部署 ID 和生putLifecycleEventHookExecutionStatus命週期事件掛接執行 ID。

以下是用 Node.js 編寫的示例 Lambda 掛鉤函數。

```
'use strict';

const aws = require('aws-sdk');
const codedeploy = new aws.CodeDeploy({apiVersion: '2014-10-06'});

exports.handler = (event, context, callback) => {
    //Read the DeploymentId from the event payload.
    var deploymentId = event.DeploymentId;

    //Read the LifecycleEventHookExecutionId from the event payload
    var lifecycleEventHookExecutionId = event.LifecycleEventHookExecutionId;

    /*
     Enter validation tests here.
    */

    // Prepare the validation test results with the deploymentId and
    // the lifecycleEventHookExecutionId for CodeDeploy.
    var params = {
        deploymentId: deploymentId,
        lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
        status: 'Succeeded' // status can be 'Succeeded' or 'Failed'
    };

    // Pass CodeDeploy the prepared validation test results.
```

```
    codedeploy.putLifecycleEventHookExecutionStatus(params, function(err, data) {
        if (err) {
            // Validation failed.
            callback('Validation test failed');
        } else {
            // Validation succeeded.
            callback(null, 'Validation test succeeded');
        }
    });
};
```

AppSpec AWS Lambda 部署的「掛鉤」部分

主題

- AWS Lambda 部署的生命週期事件掛接清單
- 在 Lambda 函數版本部署中執行掛接的順序
- 「掛鉤」部分的結構
- 示例 Lambda「掛鉤」功能

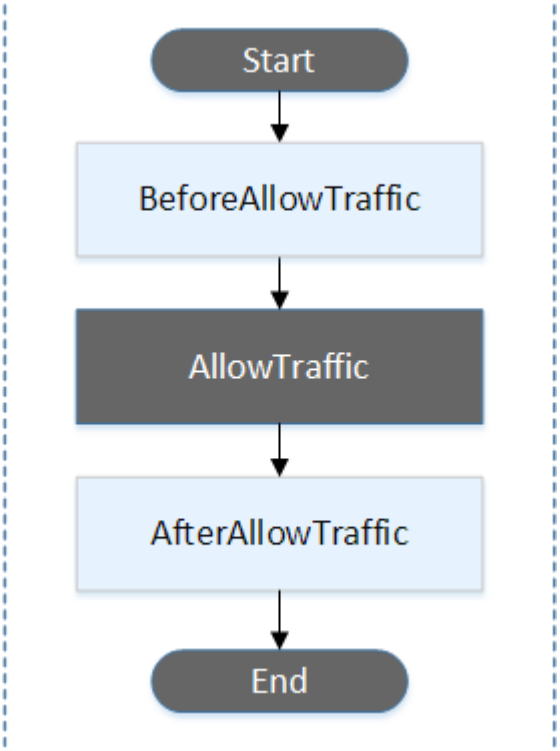
AWS Lambda 部署的生命週期事件掛接清單

AWS Lambda 掛接是一個 Lambda 函數，在生命週期事件名稱後面的新行上以字串指定。每個部署執行每個勾點一次。以下是 AppSpec 檔案中可用之掛接的描述。

- BeforeAllowTraffic— 用於在流量轉移到已部署的 Lambda 函數版本之前執行工作。
- AfterAllowTraffic— 用於在所有流量轉移到已部署的 Lambda 函數版本後執行工作。

在 Lambda 函數版本部署中執行掛接的順序

在無伺服器 Lambda 函數版本部署中，事件掛接會以下列順序執行：



> **Note**
>
> 部署中的 Start AllowTraffic、和 End 事件無法編寫指令碼，這就是為什麼它們在此圖表中以灰色顯示的原因。

「掛鉤」部分的結構

下列為 'hooks' 部分的結構範例。

使用 YAML：

```
hooks:
   - BeforeAllowTraffic: BeforeAllowTrafficHookFunctionName
   - AfterAllowTraffic: AfterAllowTrafficHookFunctionName
```

使用 JSON：

```
"hooks": [{
    "BeforeAllowTraffic": "BeforeAllowTrafficHookFunctionName"
    },
```

```
    {
    "AfterAllowTraffic": "AfterAllowTrafficHookFunctionName"
}]
```

示例 Lambda「掛鉤」功能

使用「掛接」部分指定 CodeDeploy 可呼叫以驗證 Lambda 部署的 Lambda 函數。您可以針對`BeforeAllowTraffic`和`AfterAllowTraffic`部署生命週期事件使用相同或不同的函數。完成驗證測試後，Lambda 驗證函數會回呼 CodeDeploy 並提供`Succeeded`或的結果`Failed`。

> **Important**
>
> 如果 CodeDeploy Lambda 驗證函數在一小時內未收到通知，則會將部署視為已失敗。

在叫用 Lambda 勾點函數之前，必須使用命令通知伺服器部署 ID 和生`putLifecycleEventHookExecutionStatus`命週期事件掛接執行 ID。

以下是用 Node.js 編寫的示例 Lambda 掛鉤函數。

```
'use strict';

const aws = require('aws-sdk');
const codedeploy = new aws.CodeDeploy({apiVersion: '2014-10-06'});

exports.handler = (event, context, callback) => {
    //Read the DeploymentId from the event payload.
    var deploymentId = event.DeploymentId;

    //Read the LifecycleEventHookExecutionId from the event payload
    var lifecycleEventHookExecutionId = event.LifecycleEventHookExecutionId;

    /*
     Enter validation tests here.
    */

    // Prepare the validation test results with the deploymentId and
    // the lifecycleEventHookExecutionId for CodeDeploy.
    var params = {
        deploymentId: deploymentId,
        lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
        status: 'Succeeded' // status can be 'Succeeded' or 'Failed'
```

```
    };

    // Pass CodeDeploy the prepared validation test results.
    codedeploy.putLifecycleEventHookExecutionStatus(params, function(err, data) {
        if (err) {
            // Validation failed.
            callback('Validation test failed');
        } else {
            // Validation succeeded.
            callback(null, 'Validation test succeeded');
        }
    });
};
```

AppSpec EC2 /內部部署的「掛鉤」部分

主題

- 生命週期事件掛接清單
- 生命週期事件掛接可
- 部署中掛接的執行順序
- 「掛鉤」部分的結構
- 在鉤子腳本中引用文件
- 掛接的環境變數可用性
- 掛鉤的例子

生命週期事件掛接清單

每個執行個體的部署都會執行一次 EC2/內部部署勾點。您可以指定一或多個指令碼以在勾點中執行。生命週期事件的每個勾點是在不同行上以字串指定。以下是 AppSpec 檔案中可用之掛接的描述。

如需有關哪些生命週期事件勾點對於哪些部署和復原類型有效的詳細資訊，請參閱生命週期事件掛接可。

- `ApplicationStop`— 此部署生命週期事件發生在應用程式修訂版本下載之前。在準備部署時，您可以指定這個事件的指令碼從容地停止應用程式，或移除目前已安裝的套件。用於此部署生命週期事件的 AppSpec 檔案和指令碼來自先前成功部署的應用程式修訂版本。

> **Note**
>
> 在部署到執行個體之前， AppSpec 檔案不存在於執行個體上。因此，ApplicationStop 勾點不會在您第一次部署到執行個體時執行。您可以在第二次部署到執行個體時使用 ApplicationStop 勾點。

為了確定上次成功部署的應用程式修訂版本的位置， CodeDeploy 代理程式會查詢*deployment-group-id*_last_successful_install檔案中列出的位置。此檔案位於：

/opt/codedeploy-agent/deployment-root/deployment-instructionsAmazon Linux，Ubuntu 服務器和 RHEL Amazon EC2 實例上的文件夾。

C:\ProgramData\Amazon\CodeDeploy\deployment-instructions視窗伺服器 Amazon EC2 實例上的資料夾。

若要排除部署在 ApplicationStop 部署生命週期事件期間失敗的問題，請參閱疑難排解失敗 ApplicationStop BeforeBlockTraffic、或 AfterBlockTraffic 部署生命週期事件。

- DownloadBundle— 在此部署生命週期事件期間， CodeDeploy 代理程式會將應用程式修訂版檔案複製到暫存位置：

  /opt/codedeploy-agent/deployment-root/*deployment-group-id*/*deployment-id*/deployment-archiveAmazon Linux，Ubuntu 服務器和 RHEL Amazon EC2 實例上的文件夾。

  C:\ProgramData\Amazon\CodeDeploy*deployment-group-id**deployment-id*\deployment-archive視窗伺服器 Amazon EC2 實例上的資料夾。

  此事件會保留給 CodeDeploy 代理程式使用，無法用來執行指令碼。

  若要排除部署在 DownloadBundle 部署生命週期事件期間失敗的問題，請參閱疑難排解失敗的 DownloadBundle 部署生命週期事件 UnknownError：未開啟以供讀取。

- BeforeInstall— 您可以將此部署生命週期事件用於預先安裝工作，例如解密檔案和建立目前版本的備份。
- Install— 在此部署生命週期事件期間， CodeDeploy代理程式會將修訂檔案從暫存位置複製到最終目的地資料夾。此事件會保留給 CodeDeploy 代理程式使用，無法用來執行指令碼。
- AfterInstall— 您可以將此部署生命週期事件用於設定應用程式或變更檔案權限等工作。

- `ApplicationStart`— 您通常會使用此部署生命週期事件來重新啟動期間停止的服務`ApplicationStop`。
- `ValidateService`— 這是最後一個部署生命週期事件。這用於驗證部署已順利完成。
- `BeforeBlockTraffic`— 您可以使用此部署生命週期事件，在從負載平衡器取消註冊執行個體之前，在執行個體上執行工作。

  若要排除部署在 `BeforeBlockTraffic` 部署生命週期事件期間失敗的問題，請參閱疑難排解失敗 ApplicationStop BeforeBlockTraffic、或 AfterBlockTraffic 部署生命週期事件。
- `BlockTraffic`— 在此部署生命週期事件期間，網際網路流量會遭到封鎖，無法存取目前提供流量的執行個體 此事件會保留給 CodeDeploy 代理程式使用，無法用來執行指令碼。
- `AfterBlockTraffic`— 您可以使用此部署生命週期事件，在執行個體從各自的負載平衡器取消註冊之後，在執行個體上執行工作。

  若要排除部署在 `AfterBlockTraffic` 部署生命週期事件期間失敗的問題，請參閱疑難排解失敗 ApplicationStop BeforeBlockTraffic、或 AfterBlockTraffic 部署生命週期事件。
- `BeforeAllowTraffic`— 您可以使用此部署生命週期事件，在向負載平衡器註冊執行個體之前在執行個體上執行工作。
- `AllowTraffic`— 在此部署生命週期事件期間，網際網路流量可在部署後存取執行個體。此事件會保留給 CodeDeploy 代理程式使用，無法用來執行指令碼。
- `AfterAllowTraffic`— 您可以使用此部署生命週期事件，在向負載平衡器註冊執行個體之後在執行個體上執行工作。

生命週期事件掛接可

下表列出生命週期事件勾點可用於每一個部署及復原案例。

生命週期事件名稱	自動擴展啟動部署[1]	Auto Scaling 終止部署[1]	就地部署[2]	藍/綠部署：原始執行個體	藍/綠部署：取代執行個體	藍/綠部署復原：原始執行個體	藍/綠部署復原：取代執行個體
ApplicationStop	✓	✓	✓		✓		
DownloadBundle[3]	✓		✓		✓		

生命週期事件名稱	自動擴展啟動部署[1]	Auto Scaling 終止部署[1]	就地部署[2]	藍/綠部署：原始執行個體	藍/綠部署：取代執行個體	藍/綠部署復原：原始執行個體	藍/綠部署復原：取代執行個體
BeforeInstall	✓		✓		✓		
安裝[3]	✓		✓		✓		
AfterInstall	✓		✓		✓		
ApplicationStart	✓		✓		✓		
ValidateService	✓		✓		✓		
BeforeBlockTraffic		✓	✓	✓			✓
BlockTraffic[3]		✓	✓	✓			✓
AfterBlockTraffic		✓	✓	✓			✓
BeforeAllowTraffic	✓		✓		✓	✓	
AllowTraffic[3]	✓		✓		✓	✓	
AfterAllowTraffic	✓		✓		✓	✓	

生命週期事件名稱	自動擴展啟動部署[1]	Auto Scaling 終止部署[1]	就地部署[2]	藍/綠部署：原始執行個體	藍/綠部署：取代執行個體	藍/綠部署復原：原始執行個體	藍/綠部署復原：取代執行個體

[1] 如需 Amazon EC2 Auto Scaling 部署的相關資訊，請參閱亞馬遜 EC2 Auto Scaling 如何與 CodeDeploy。

[2] 也適用於就地部署的復原。

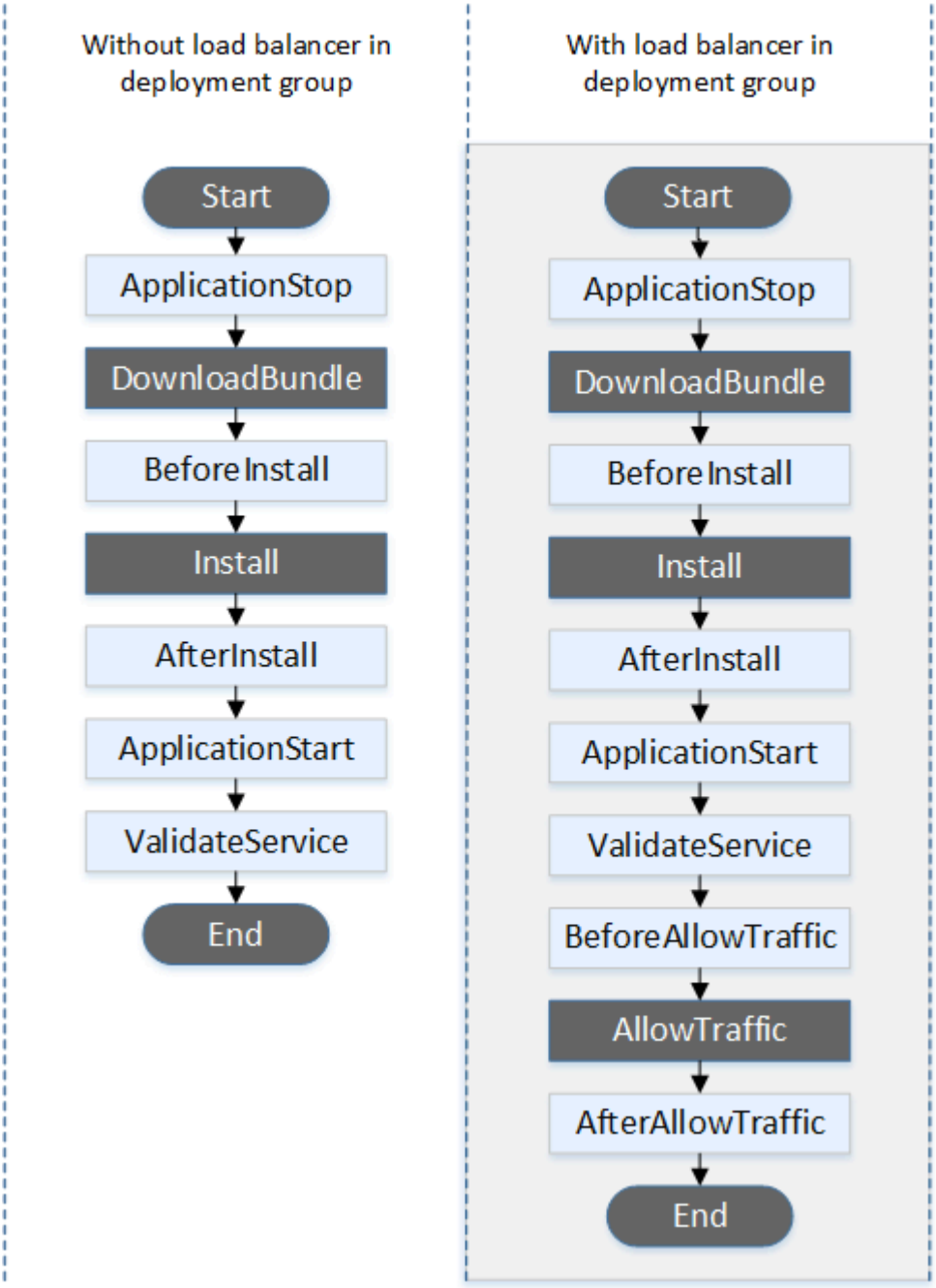
[3] 保留用於 CodeDeploy 操作。無法用於執行指令碼。

部署中掛接的執行順序

Auto Scaling 啟動部署

在 Auto Scaling 啟動部署期間， CodeDeploy 會依下列順序執行事件掛接。

如需 Auto Scaling 啟動部署的詳細資訊，請參閱亞馬遜 EC2 Auto Scaling 如何與 CodeDeploy。

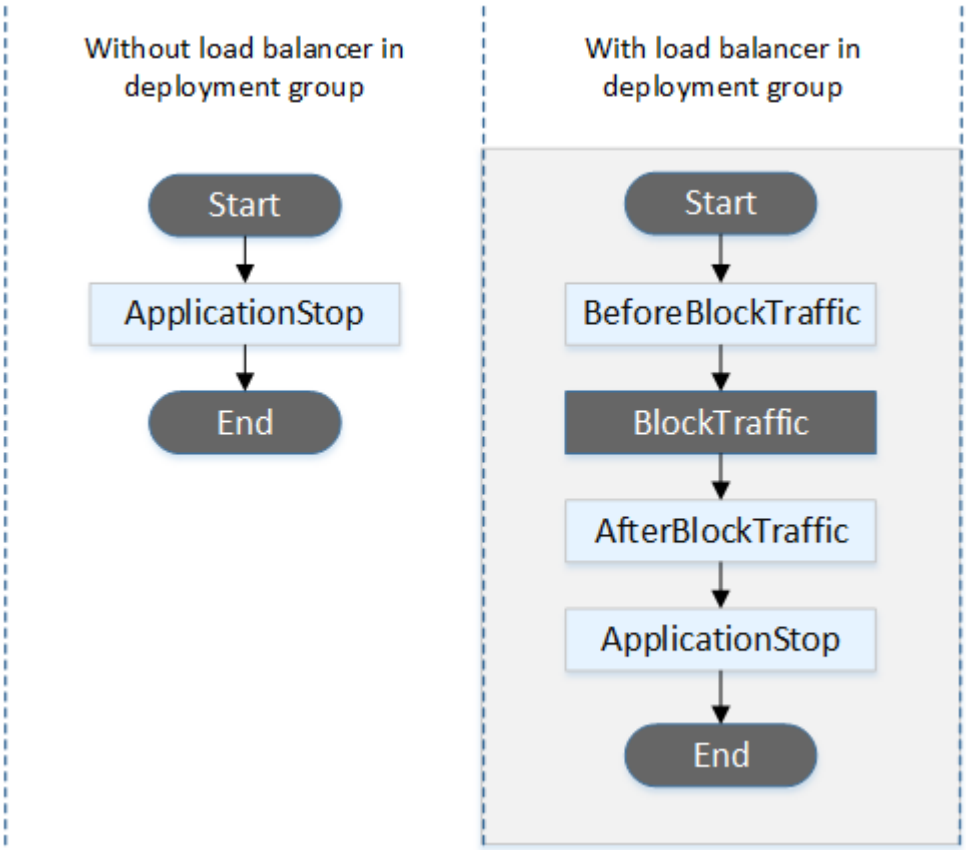


> **Note**
>
> 部署中的「開始」AllowTraffic、「安裝」和「結束」事件無法編寫指令碼，這就是為什麼它們在此圖表中以灰色顯示的原因。DownloadBundle不過，您可以編輯 AppSpec 檔案的`'files'`區段，以指定在 Install 事件期間安裝的項目。

Auto Scaling 終止部署

在 Auto Scaling 終止部署期間， CodeDeploy 會依下列順序執行事件掛接。

如需 Auto Scaling 終止部署的詳細資訊，請參閱在 Auto Scaling 擴充事件期間啟用終止部署。

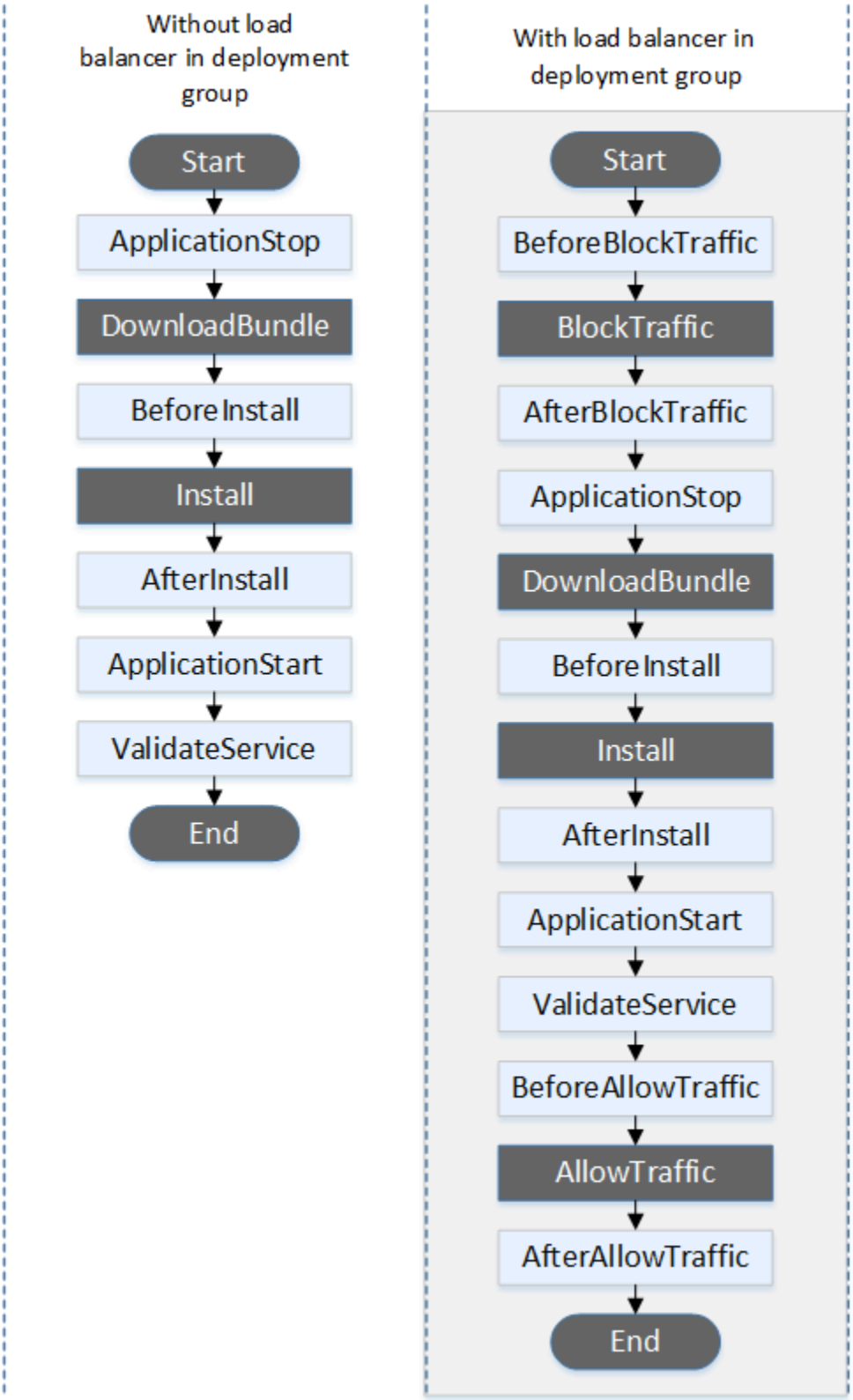


> **Note**
>
> 部署中的 Start BlockTraffic、和 End 事件無法編寫指令碼，這就是為什麼它們在此圖表中以灰色顯示的原因。

就地部署：

在就地部署中，包括就地部署的復原，事件勾點以下列順序執行：

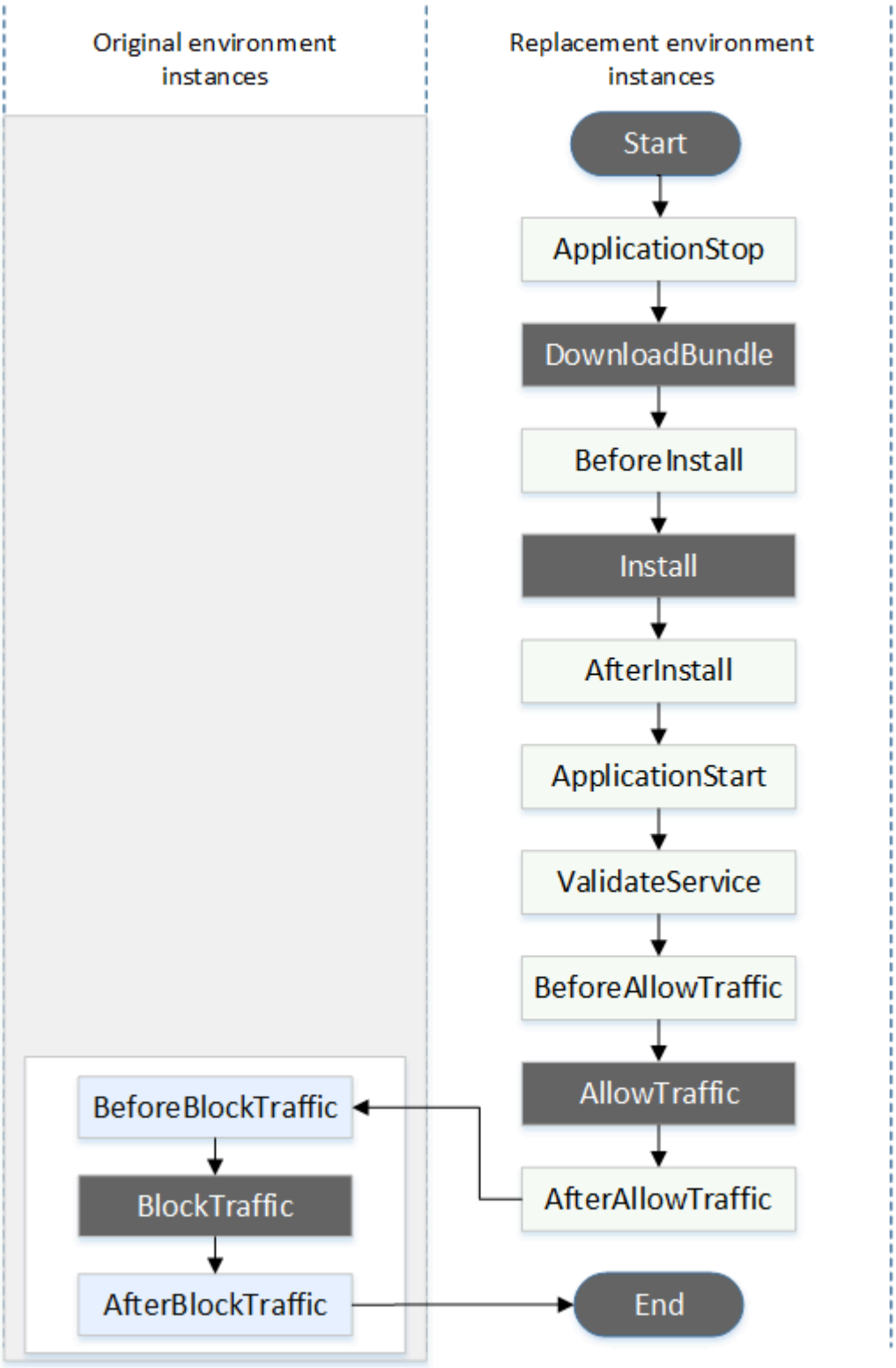
> **Note**
>
> 對於就地部署，只有當您在部署群組中從 Elastic Load Balancing 指定 Classic Load Balancer、應用程式負載平衡器或 Network Load Balancer 時，才會套用與封鎖和允許流量相關的六個掛接。


Without load balancer in deployment group
Start
ApplicationStop
DownloadBundle
BeforeInstall
Install
AfterInstall
ApplicationStart
ValidateService
End
With load balancer in deployment group
Start
BeforeBlockTraffic
BlockTraffic
AfterBlockTraffic
ApplicationStop
DownloadBundle
BeforeInstall
Install
AfterInstall
ApplicationStart
ValidateService
BeforeAllowTraffic
AllowTraffic
AfterAllowTraffic
End

> **Note**
>
> 部署中的「開始」DownloadBundle、「安裝」和「結束」事件無法編寫指令碼，這就是為什麼它們在此圖表中以灰色顯示的原因。不過，您可以編輯 AppSpec 檔案的`'files'`區段，以指定在 Install 事件期間安裝的項目。

藍/綠部署

在藍/綠部署中，事件勾點以下列順序執行：


Original environment instances
Replacement environment instances
Start
ApplicationStop
DownloadBundle
BeforeInstall
Install
AfterInstall
ApplicationStart
ValidateService
BeforeAllowTraffic
AllowTraffic
AfterAllowTraffic
BeforeBlockTraffic
BlockTraffic
AfterBlockTraffic
End

> **Note**
>
> 部署中的「開始」DownloadBundle、「安裝」BlockTrafficAllowTraffic、「」和「結束」事件無法編寫指令碼，這就是為什麼它們在此圖表中以灰色顯示的原因。但是，您可以編輯檔案的「 AppSpec 檔案」區段，以指定在 Install 事件期間安裝的內容。

「掛鉤」部分的結構

'hooks' 區段的結構如下：

```
hooks:
   deployment-lifecycle-event-name:
     - location: script-location
       timeout: timeout-in-seconds
       runas: user-name
```

您可以在 hook 項目中，在部署生命週期事件名稱後包含下列元素：

location

必要。修訂版指令碼檔案的套件組合位置。您在此hooks段落中指定的程序檔位置是相對於應用程式修訂版套裝軟體的根目錄。如需詳細資訊，請參閱 規劃修訂 CodeDeploy。

timeout

選用。指令碼被視為失敗前允許執行的秒數。預設為 3600 秒 (1 小時)。

> **Note**
>
> 3600秒 (1小時) 是允許為每個部署生命週期事件執行指令碼時間上限。如果指令碼超過此限制，部署會停止，且部署到執行個體會失敗。請確認每一個部署生命週期事件內所有的指令碼所指定的 timeout 總秒數不超過這個限制。

runas

選用。當執行指令碼時要模擬的使用者。依預設，這是執行個體上執行的 CodeDeploy 代理程式。 CodeDeploy 不存儲密碼，因此如果 runas 用戶需要密碼，則無法模擬用戶。此元素僅適用於 Amazon Linux 和 Ubuntu 伺服器執行個體。

在鉤子腳本中引用文件

如果您要按照中所述將腳本連接到 CodeDeploy 生命週期事件AppSpec 「掛鉤」部分，並且想要引用腳本中的文件（例如，`helper.sh`），則需要`helper.sh`使用以下命令來指定：

- (建議使用) 絕對路徑。請參閱使用絕對路徑。
- 相對路徑。請參閱使用相對路徑。

使用絕對路徑

若要使用其絕對路徑參照檔案，您可以：

- 在`destination`屬性的 AppSpec 檔案`files`區段中指定絕對路徑。然後，在鉤子腳本中指定相同的絕對路徑。如需詳細資訊，請參閱 AppSpec 「檔案」區段 (僅適用於 EC2 /內部部署)。
- 在鉤子腳本中指定動態絕對路徑。如需詳細資訊，請參閱部署封存位置。

部署封存位置

在DownloadBundle生命週期事件期間， CodeDeploy 代理程式會將部署的修訂擷取至具有下列格式的目錄：

root-directory/*deployment-group-id*/*deployment-id*/deployment-archive

路徑的*###*部分一律設定為下表所示的預設值，或由`:root_dir`組態設定控制。如需組態設定的詳細資訊，請參閱CodeDeploy 用戶端組態參考。

代理平台	預設根目錄
Linux — 所有轉速發行版	`/opt/codedeploy-agent/deployment-root`
Ubuntu 的服務器-所有 DEB 發行版	`/opt/codedeploy-agent/deployment-root`
Windows Server	`%ProgramData%\Amazon\CodeDeploy`

從勾點指令碼中，您可以使用根目錄路徑和和DEPLOYMENT_GROUP_ID環境變數來存取目前的DEPLOYMENT_ID部署封存檔。如需有關可使用之變數的更多資訊，請參閱掛接的環境變數可用性。

例如，以下是如何在 Linux 上訪問位於修訂根目錄的data.json文件：

```
#!/bin/bash

rootDirectory="/opt/codedeploy-agent/deployment-root" # note: this will be different if
 you
                                                      # customize the :root_dir
 configuration
dataFile="$rootDirectory/$DEPLOYMENT_GROUP_ID/$DEPLOYMENT_ID/deployment-archive/
data.json"
data=$(cat dataFile)
```

作為另一個例子，以下是如何使用 Windows 上的 Powershell 訪問版本根目錄中的data.json文件：

```
$rootDirectory="$env:ProgramData\Amazon\CodeDeploy" # note: this will be different if
 you
                                                    # customize the :root_dir
 configuration
$dataFile="$rootDirectory\$env:DEPLOYMENT_GROUP_ID\$env:DEPLOYMENT_ID\deployment-
archive\data.json"
$data=(Get-Content $dataFile)
```

使用相對路徑

若要使用其相對路徑來參考檔案，您需要知道 CodeDeploy 代理程式的工作目錄。檔案路徑是相對於此目錄的路徑。

下表顯示 CodeDeploy 代理程式每個支援平台的工作目錄。

代理平台	流程管理方法	生命週期事件腳本的工作目錄
Linux — 所有轉速發行版	系統 (預設值)	/
	init.d — 瞭解更多資訊	/opt/codedeploy-agent
Ubuntu 伺服器 — 所有 Debian 發行版	全部	/opt/codedeploy-agent

代理平台	流程管理方法	生命週期事件腳本的工作目錄
Windows Server	不適用	C:\Windows\System32

掛接的環境變數可用性

在每個部署生命週期事件期間，勾點指令碼能存取以下環境變數：

APPLICATION_NAME

屬於目前部署 CodeDeploy 一部分的應用程式名稱 (例如WordPress_App)。

DEPLOYMENT_ID

ID CodeDeploy 已指派給目前部署 (例如d-AB1CDEF23)。

DEPLOYMENT_GROUP_NAME

屬於目前部署一部分 CodeDeploy 的部署群組名稱 (例如WordPress_DepGroup)。

DEPLOYMENT_GROUP_ID

屬於目前部署一部分 CodeDeploy 的部署群組識別碼 (例如b1a2189b-dd90-4ef5-8f40-4c1c5EXAMPLE)。

LIFECYCLE_EVENT

目前部署生命週期事件的名稱 (例如，AfterInstall)。

這些環境變數在每個部署生命週期事件的本機。

根據部署服務包的來源，還有其他可用於掛接指令碼的環境變數：

來自 Amazon S3 的捆綁

- 捆綁桶

  從中下載部署服務包的 Amazon S3 儲存貯體的名稱 (例如my-s3-bucket)。

- 捆綁密鑰

  Amazon S3 儲存貯體中已下載服務包的物件金鑰 (例如，WordPress_App.zip)。

- 捆綁版本

套裝軟體的物件版本 (例如，3sL4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY+MTRCxf3vjVBH40Nr8X8gdRQBpUMLUo)。只有在 Amazon S3 儲存貯體已啟用物件版本控制時，才會設定此變數。

- 捆綁電子標籤

  套裝軟體的物件 etag (例如，b10a8db164e0754105b7a99be72e3fe5-4)。

捆綁來源 GitHub

- 捆綁提交

  由 Git 生成的包的 SHA256 提交哈希值（例如，d2a84f4b8b650937ec8f73cd8be2c74add5a911ba64df27458ed8229da804a26）。

如果 DEPLOYMENT_GROUP_NAME 的值等於 Staging，以下指令碼會變更Apache HTTP 伺服機上的接聽連接埠到 9090 以代替 80。這個指令碼必須在 BeforeInstall 部署生命週期事件期間叫用：

```
if [ "$DEPLOYMENT_GROUP_NAME" == "Staging" ]
then
    sed -i -e 's/Listen 80/Listen 9090/g' /etc/httpd/conf/httpd.conf
fi
```

如果 DEPLOYMENT_GROUP_NAME 環境變數的值等於 Staging，以下指令碼範例會將錯誤日誌中所記錄的詳細資訊等級，從警告變更為偵錯。這個指令碼必須在 BeforeInstall 部署生命週期事件期間叫用：

```
if [ "$DEPLOYMENT_GROUP_NAME" == "Staging" ]
then
    sed -i -e 's/LogLevel warn/LogLevel debug/g' /etc/httpd/conf/httpd.conf
fi
```

以下指令碼範例取代被指定的網頁文字，其顯示這些環境變數的值。這個指令碼必須在 AfterInstall 部署生命週期事件期間叫用：

```
#!/usr/bin/python

import os
```

```
strToSearch="<h2>This application was deployed using CodeDeploy.</h2>"
strToReplace="<h2>This page for "+os.environ['APPLICATION_NAME']+"
 application and "+os.environ['DEPLOYMENT_GROUP_NAME']+" deployment group with
 "+os.environ['DEPLOYMENT_GROUP_ID']+" deployment group ID was generated by a
 "+os.environ['LIFECYCLE_EVENT']+" script during "+os.environ['DEPLOYMENT_ID']+"
 deployment.</h2>"

fp=open("/var/www/html/index.html","r")
buffer=fp.read()
fp.close()

fp=open("/var/www/html/index.html","w")
fp.write(buffer.replace(strToSearch,strToReplace))
fp.close()
```

掛鉤的例子

這裡有一個 hooks 輸入項目的範例，其為 `AfterInstall` 生命週期事件指定兩個勾點。

```
hooks:
   AfterInstall:
     - location: Scripts/RunResourceTests.sh
       timeout: 180
     - location: Scripts/PostDeploy.sh
       timeout: 180
```

`Scripts/RunResourceTests.sh`指令碼在部署程序的 `AfterInstall` 階段期間執行。如果它需要超過 180 秒 (3分鐘) 去執行指令碼，則表示部署是不成功的。

您在 'hooks' 部分指定的指令碼位置，與應用程式修訂版套件組合的根目錄相關。在前述案例中，名為 `RunResourceTests.sh` 的檔案位於名為 `Scripts` 的目錄中。`Scripts` 目錄位於套件組合的根層級。如需詳細資訊，請參閱 規劃修訂 CodeDeploy。

AppSpec 檔案範例

本主題提供 L AWS ambda 和 EC2 /內部部署的範例檔 AppSpec 案。

主題

- AppSpec Amazon ECS 部署的檔案範例
- AppSpec AWS Lambda 部署的檔案範例

- AppSpec EC2/內部部署的檔案範例

AppSpec Amazon ECS 部署的檔案範例

以下是使用 YAML 撰寫用來部署 Amazon ECS 服務的 AppSpec 檔案範例。

```
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "arn:aws:ecs:us-east-1:111222333444:task-definition/my-task-
definition-family-name:1"
        LoadBalancerInfo:
          ContainerName: "SampleApplicationName"
          ContainerPort: 80
# Optional properties
        PlatformVersion: "LATEST"
        NetworkConfiguration:
          AwsvpcConfiguration:
            Subnets: ["subnet-1234abcd","subnet-5678abcd"]
            SecurityGroups: ["sg-12345678"]
            AssignPublicIp: "ENABLED"
        CapacityProviderStrategy:
          - Base: 1
            CapacityProvider: "FARGATE_SPOT"
            Weight: 2
          - Base: 0
            CapacityProvider: "FARGATE"
            Weight: 1
Hooks:
  - BeforeInstall: "LambdaFunctionToValidateBeforeInstall"
  - AfterInstall: "LambdaFunctionToValidateAfterInstall"
  - AfterAllowTestTraffic: "LambdaFunctionToValidateAfterTestTrafficStarts"
  - BeforeAllowTraffic: "LambdaFunctionToValidateBeforeAllowingProductionTraffic"
  - AfterAllowTraffic: "LambdaFunctionToValidateAfterAllowingProductionTraffic"
```

這裡提供前述以 JSON 撰寫的範例版本。

```
{
    "version": 0.0,
    "Resources": [
```

```
        {
            "TargetService": {
                "Type": "AWS::ECS::Service",
                "Properties": {
                    "TaskDefinition": "arn:aws:ecs:us-east-1:111222333444:task-
definition/my-task-definition-family-name:1",
                    "LoadBalancerInfo": {
                        "ContainerName": "SampleApplicationName",
                        "ContainerPort": 80
                    },
                    "PlatformVersion": "LATEST",
                    "NetworkConfiguration": {
                        "AwsvpcConfiguration": {
                            "Subnets": [
                                "subnet-1234abcd",
                                "subnet-5678abcd"
                            ],
                            "SecurityGroups": [
                                "sg-12345678"
                            ],
                            "AssignPublicIp": "ENABLED"
                        }
                    },
                    "CapacityProviderStrategy": [
                        {
                            "Base" : 1,
                            "CapacityProvider" : "FARGATE_SPOT",
                            "Weight" : 2
                        },
                        {
                            "Base" : 0,
                            "CapacityProvider" : "FARGATE",
                            "Weight" : 1
                        }
                    ]
                }
            }
        }
    ],
    "Hooks": [
        {
            "BeforeInstall": "LambdaFunctionToValidateBeforeInstall"
        },
        {
```

```
            "AfterInstall": "LambdaFunctionToValidateAfterInstall"
        },
        {
            "AfterAllowTestTraffic": "LambdaFunctionToValidateAfterTestTrafficStarts"
        },
        {
            "BeforeAllowTraffic":
 "LambdaFunctionToValidateBeforeAllowingProductionTraffic"
        },
        {
            "AfterAllowTraffic":
 "LambdaFunctionToValidateAfterAllowingProductionTraffic"
        }
    ]
}
```

這是部署期間的一系列事件。

1. 在替換任務集上安裝更新的 Amazon ECS 應用程式之前，名為的 Lambda 函數`LambdaFunctionToValidateBeforeInstall`會執行。
2. 在替換任務集上安裝更新的 Amazon ECS 應用程式之後，但在接收任何流量之前，稱為的 Lambda 函數會`LambdaFunctionToValidateAfterInstall`執行。
3. 取代任務集上的 Amazon ECS 應用程式開始接收來自測試接聽程式的流量之後，稱為 Lambda 函數即會`LambdaFunctionToValidateAfterTestTrafficStarts`執行。此函數可能執行驗證測試，判斷是否繼續部署。如果您未指定測試部署群組中的測試接聽程式，便會忽略此勾點。
4. `AfterAllowTestTraffic`勾點中的任何驗證測試完成後，在將生產流量提供給更新的 Amazon ECS 應用程式之前，稱為的 Lambda 函數會`LambdaFunctionToValidateBeforeAllowingProductionTraffic`執行。
5. 在替換任務集上將生產流量提供給更新的 Amazon ECS 應用程式之後，稱為的 Lambda 函數便`LambdaFunctionToValidateAfterAllowingProductionTraffic`會執行。

在任何掛接期間執行的 Lambda 函數都可以執行驗證測試或收集流量指標。

AppSpec AWS Lambda 部署的檔案範例

以下是使用 YAML 撰寫的 AppSpec 檔案範例，用來部署 Lambda 函數版本。

```
version: 0.0
Resources:
```

```
  - myLambdaFunction:
      Type: AWS::Lambda::Function
      Properties:
        Name: "myLambdaFunction"
        Alias: "myLambdaFunctionAlias"
        CurrentVersion: "1"
        TargetVersion: "2"
Hooks:
  - BeforeAllowTraffic: "LambdaFunctionToValidateBeforeTrafficShift"
  - AfterAllowTraffic: "LambdaFunctionToValidateAfterTrafficShift"
```

這裡提供前述以 JSON 撰寫的範例版本。

```
{
  "version": 0.0,
  "Resources": [{
   "myLambdaFunction": {
    "Type": "AWS::Lambda::Function",
    "Properties": {
     "Name": "myLambdaFunction",
     "Alias": "myLambdaFunctionAlias",
     "CurrentVersion": "1",
     "TargetVersion": "2"
    }
   }
  }],
  "Hooks": [{
    "BeforeAllowTraffic": "LambdaFunctionToValidateBeforeTrafficShift"
      },
      {
    "AfterAllowTraffic": "LambdaFunctionToValidateAfterTrafficShift"
   }
  ]
 }
```

這是部署期間的一系列事件。

1. 在將流量從呼叫的 Lambda 函數的第 1 版轉移myLambdaFunction到第 2 版之前，請執行名為的 Lambda 函數LambdaFunctionToValidateBeforeTrafficShift，以驗證部署是否準備好開始流量轉移。
2. 如果 LambdaFunctionToValidateBeforeTrafficShift 傳回結束代碼 0 (成功)，請開始轉移流量到第 2 版的 myLambdaFunction。此部署的部署組態決定了流量轉移速率。

3. 將流量從呼叫的 Lambda 函數第 1 版轉移`myLambdaFunction`到第 2 版之後，請執行名為的 Lambda 函數，`LambdaFunctionToValidateAfterTrafficShift`以驗證部署是否已順利完成。

AppSpec EC2/內部部署的檔案範例

以下是用於就地部署到 Amazon Linux、Ubuntu 伺服器或 RHEL 執行個體的 AppSpec 檔案範例。

> **Note**
>
> 對 Windows 伺服器執行個體的部署不支援該`runas`元素。如果您要部署到 Windows 伺服器執行個體，請勿將其包含在您的 AppSpec 檔案中。

```
version: 0.0
os: linux
files:
  - source: Config/config.txt
    destination: /webapps/Config
  - source: source
    destination: /webapps/myApp
hooks:
  BeforeInstall:
    - location: Scripts/UnzipResourceBundle.sh
    - location: Scripts/UnzipDataBundle.sh
  AfterInstall:
    - location: Scripts/RunResourceTests.sh
      timeout: 180
  ApplicationStart:
    - location: Scripts/RunFunctionalTests.sh
      timeout: 3600
  ValidateService:
    - location: Scripts/MonitorService.sh
      timeout: 3600
      runas: codedeployuser
```

如果是 Windows 伺服器執行個體，請變更`os: linux`為`os: windows`。而且，您必須有完整的 `destination` 路徑 (例如，`c:\temp\webapps\Config` 和 `c:\temp\webapps\myApp`)。請勿包含 `runas` 元素。

這是部署期間的一系列事件。

1. 執行位於 Scripts/UnzipResourceBundle.sh 的指令碼。
2. 如果之前的指令碼傳回 0 結束代碼 (成功)，請執行位於 Scripts/UnzipDataBundle.sh 的指令碼。
3. 從 Config/config.txt 的路徑複製檔案到 /webapps/Config/config.txt 路徑。
4. 以遞迴方式複製 source 目錄中的所有檔案到 /webapps/myApp 目錄。
5. 以 180 秒 (3分鐘) 的逾時時間於 Scripts/RunResourceTests.sh 執行指令碼。
6. 以 3600 秒 (1 小時) 的逾時時間於 Scripts/RunFunctionalTests.sh 執行指令碼。
7. 以使用者 codedeploy 身分於 3600 秒 (1小時) 的逾時時間執行位於 Scripts/MonitorService.sh 的指令碼。

AppSpec 檔案間距

以下是 AppSpec 檔案間距的正確格式。方格括弧中的數字表示在項目之間必需的空格數。例如，[4]意味著在項目之間插入四個空格。 CodeDeploy 如果檔案中的位置和空格數目不正確，則會引發可能難以偵 AppSpec 錯的錯誤。

```
version:[1]version-number
os:[1]operating-system-name
files:
[2]-[1]source:[1]source-files-location
[4]destination:[1]destination-files-location
permissions:
[2]-[1]object:[1]object-specification
[4]pattern:[1]pattern-specification
[4]except:[1]exception-specification
[4]owner:[1]owner-account-name
[4]group:[1]group-name
[4]mode:[1]mode-specification
[4]acls:
[6]-[1]acls-specification
[4]context:
[6]user:[1]user-specification
[6]type:[1]type-specification
[6]range:[1]range-specification
[4]type:
[6]-[1]object-type
hooks:
[2]deployment-lifecycle-event-name:
[4]-[1]location:[1]script-location
```

```
[6]timeout:[1]timeout-in-seconds
[6]runas:[1]user-name
```

下面是一個正確間隔的 AppSpec 文件的例子：

```
version: 0.0
os: linux
files:
  - source: /
    destination: /var/www/html/WordPress
hooks:
  BeforeInstall:
    - location: scripts/install_dependencies.sh
      timeout: 300
      runas: root
  AfterInstall:
    - location: scripts/change_permissions.sh
      timeout: 300
      runas: root
  ApplicationStart:
    - location: scripts/start_server.sh
    - location: scripts/create_test_db.sh
      timeout: 300
      runas: root
  ApplicationStop:
    - location: scripts/stop_server.sh
      timeout: 300
      runas: root
```

如需有關間距的詳細資訊，請參閱 YAML 規格。

驗證您的 AppSpec 檔案和檔案位置

檔案語法

您可以使用 AWS提供的 AppSpec 助理腳本來驗證 AppSpec 文件的內容。您可以在上找到腳本以及 AppSpec 文件模板GitHub。

您也可以使用以瀏覽器為基礎的工具，例如 YAML lint 或線上 YAML 剖析器來協助您檢查 YAML 語法。

檔案位置

若要確認您已將 AppSpec 檔案置於應用程式來源內容目錄結構的根目錄中，請執行下列其中一個指令：

在本機 Linux、macOS 或 Unix 執行個體上：

```
ls path/to/root/directory/appspec.yml
```

如果該 AppSpec 文件不在該處，則顯示「沒有此類文件或目錄」錯誤。

在本機 Windows 執行個體：

```
dir path\to\root\directory\appspec.yml
```

如果 AppSpec 檔案未找到該檔案，則會顯示「找不到檔案」錯誤。

CodeDeploy 用戶端組態參考

安裝 CodeDeploy 代理程式後，系統會在執行個體上放置組態檔案。此組態檔指定目錄路徑和其他設定， CodeDeploy 以便在與執行個體互動時使用。您可以在檔案中變更一部分的組態選項。

對於 Amazon Linux、Ubuntu 伺服器和 RHEL (RHEL) 執行個體，組態檔案會被命名`codedeployagent.yml`為。它會置放於 `/etc/codedeploy-agent/conf` 目錄中。

對於 Windows 伺服器執行個體，組態檔案會被命名為`conf.yml`。它會置放於 `C:\ProgramData\Amazon\CodeDeploy` 目錄中。

組態設定包含：

:log_aws_wire:	將 CodeDeploy 代理程式設定**`true`**為以從 Amazon S3 擷取線日誌，並將其寫入由:log_dir: 設定指向的位置命**`codedeploy-agent.wire.log`** 名的檔案。 **Warning** 您只應該針對擷取線路日誌所需時間量，將 :log_aws_wire: 設定為 `true`。`codedeploy-agent.wire.log` 檔案大小可能會變很大。

	此檔案中的電線日誌輸出可能包含敏感資訊，包括在此設定設為時傳入或傳出 Amazon S3 的檔案的純文字內容。`true`線日誌包含與此設定設定為時 AWS 帳戶相關聯的所有 Amazon S3 活動的相關資訊`true`，而不僅僅是與 CodeDeploy 部署相關的活動。 預設設定為 `false`。 此設定適用於所有執行個體類型。您必須將此組態設定新增至 Windows 伺服器執行個體，才能使用它。
:log_dir:	執行個體上儲存與 CodeDeploy 代理程式作業相關的記錄檔的資料夾。 預設設定`'/var/log/aws/codedeploy-agent'` 適用於 Amazon Linux、Ubuntu 伺服器和 RHEL 執行個體，以及`C:\ProgramData\Amazon\CodeDeploy\log` 適用於 Windows 伺服器執行個體。
:pid_dir:	存放 `codedeploy-agent.pid` 的資料夾。 此檔案包含代理程式的處 CodeDeploy 理程序 ID (PID)。預設設定為 `'/opt/codedeploy-agent/state/.pid'` 。 此設定僅適用於 Amazon Linux、Ubuntu 伺服器和 RHEL 執行個體。
:program_name:	CodeDeploy 代理程式程式名稱。 預設設定為 `codedeploy-agent` 。 此設定僅適用於 Amazon Linux、Ubuntu 伺服器和 RHEL 執行個體。

:root_dir:	存放執行個體上相關修訂、部署歷史記錄和部署指令碼的資料夾。 預設設定`/opt/codedeploy-agent/deployment-root` 適用於 Amazon Linux、Ubuntu 伺服器和 RHEL 執行個體，以及`C:\ProgramData\Amazon\CodeDeploy` 適用於 Windows 伺服器執行個體。
:verbose:	設定`true`為， CodeDeploy 代理程式可在執行個體上列印除錯訊息記錄檔。 預設設定為 `false`。
:wait_between_runs:	擱置部署的 CodeDeploy 代理程式輪詢之間 CodeDeploy 的間隔 (秒)。 預設設定為 1。
:on_premises_config_file:	對於內部部署執行個體，指定名為 (適用於 Ubuntu 伺服器和 RHEL) 或 `codedeploy.onpremises.yml` `conf.onpremises.yml` (適用於 Windows 伺服器) 之組態檔案的替代位置路徑。 默認情況下，這些文件存儲在`/etc/codedeploy-agent/conf` `/codedeploy.onpremises.yml` Ubuntu 服務器和 RHEL 和`C:\ProgramData\Amazon\CodeDeploy \conf.onpremises.yml` 用於 Windows 服務器。 提供代理程式版本 1.0.1.686 及更新版本。 CodeDeploy

:proxy_uri:	(選擇性) 您希望代理程 CodeDeploy 式為您的 CodeDeploy 作業連線到 AWS 的 HTTP Proxy。使用與 `https://user:password@my.proxy:443/path?query` 類似的格式。 提供代理程式版本 1.0.1.824 及更新版本。CodeDeploy
:max_revisions:	(選擇性) 您希望代理程式封存之部署群組的應用程 CodeDeploy 式修訂數目。任何超過所指定號碼的修訂都會予以刪除。 輸入任何正整數。如果未指定任何值，除了目前部署的修訂版本之外，還 CodeDeploy 會保留最新的五個修訂。 在版本 1.0.1.966 及更新版本的代理程式中支援。 CodeDeploy
：啟用政策：	(選擇性) `true` 如果您想要使用 IAM 授權來設定存取控制，並限制 CodeDeploy 代理程式所使用之 IAM 角色或使用者的權限，請設定為。若要搭 CodeDeploy 配 Amazon Virtual Private Cloud 使用，此值必須是`true`。 預設設定為 `false`。
：禁用 _ 模組 _v1：	此設定適用於 CodeDeploy 代理程式 1.7.0 及更新版本。 設定`true`為可在發生 IMDSv2 錯誤時停用 IMDSv1 的後援功能。預設為 `false` (啟用後援)。

相關主題

與 CodeDeploy 代理工作

管理 CodeDeploy 代理程式作

AWS CloudFormation CodeDeploy供參考的範本

本節介紹專為配合 CodeDeploy 部署使用而設計的 AWS CloudFormation 資源、轉換和掛接。如需建立由 AWS CloudFormation 掛接管理之堆疊更新的逐步解說 CodeDeploy，請參閱 透過以下方式建立亞馬遜ECS 藍色/綠色部署 AWS CloudFormation

> **Note**
>
> AWS CloudFormation 掛接是 AWS CloudFormation 元件的一部分， AWS 與 CodeDeploy生命週期事件掛接不同。

除了中可用的其他方法之外 CodeDeploy，您還可以使用 AWS CloudFormation 範本來執行下列工作：

- 建立應用程式。
- 建立部署群組，並指定目標修訂版。
- 建立部署組態。
- 創建 Amazon EC2 實例。

AWS CloudFormation 是一項可協助您使用範本建立資源模型和設定 AWS 資源的服務。 AWS CloudFormation 範本是格式符合 JSON 標準的文字檔案。您可以建立一個範本來描述所 AWS CloudFormation 需的所有 AWS 資源，並為您處理佈建和設定這些資源。

如需詳細資訊，請參閱什麼是 AWS CloudFormation？ 以及使AWS CloudFormation 用《使用指南》中的 AWS CloudFormation 範本。

如果您打算以管理員身分使用與組織 CodeDeploy 中相容的 AWS CloudFormation 範本，您必須授與存取權，以 AWS CloudFormation 及存取相 AWS CloudFormation 依的 AWS 服務和動作。若要授與建立應用程式、部署群組和部署組態的權限，請將下列原則新增至將使用者的權限集 AWS CloudFormation：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
```

```
        "cloudformation:*"
      ],
      "Resource": "*"
    }
  ]
}
```

如需原則的相關資訊，請參閱下列主題：

- 若要檢視必須新增至將建立 Amazon EC2 執行個體之使用者權限集的政策，請參閱為 CodeDeploy (AWS CloudFormation 範本) 建立 Amazon EC2 執行個體。
- 如需將政策新增至權限集的相關資訊，請參閱《IAM 使用者指南》中的「建立權限集」。
- 若要瞭解如何將使用者限制在一組有限的 CodeDeploy 動作和資源，請參閱AWS 的管理 (預先定義) 策略 CodeDeploy。

下表顯示 AWS CloudFormation 範本可以代表您執行的動作，並包含可新增至 AWS CloudFormation 範本的 AWS 資源類型及其屬性類型的詳細資訊連結。

動作	AWS CloudFormation 參考	參考類型
建立 CodeDeploy 應用程式。	AWS::Code部署:: 應用	AWS CloudFormation 資源
建立並指定要用來部署應用程式修訂版本之部署群組的詳細資料。[1]	AWS::CodeDeploy::DeploymentGroup	AWS CloudFormation 資源
建立部署期間 CodeDeploy 將使用的一組部署規則、部署成功條件和部署失敗條件。	AWS::CodeDeploy::DeploymentConfig	AWS CloudFormation 資源
創建一個 Amazon EC2 實例。[2]	AWS:: EC2:: 執行個體	AWS CloudFormation 資源
使用 AWS CloudFormation `AWS::CodeDeployBlueGreen` 轉換和`AWS::CodeDeploy::BlueGreen` 掛接來管理堆疊更新、建立資源，	AWS::CodeDeployBlueGreen	該 `AWS::CodeDeployBlueGreen` 轉換是由 AWS CloudFormation 託管的巨集
	AWS::CodeDeploy::BlueGreen	`AWS::CodeDeploy::BlueGreen` 掛接的結構為中

動作	AWS CloudFormation 參考	參考類型
以及移轉 CodeDeploy 藍/綠部署的流量。[3]		的Hook資源 AWS CloudFormation。掛接包括透過指向指定的生 CodeDeploy命週期事件掛接取代 CodeDeploy AppSpec 檔案的參數。

[1] 如果您指定要部署為部署群組一部分的應用程式修訂版本版本，則會在佈建程序完成後立即部署您的目標修訂版本。若要取得有關範本規劃的更多資訊，請參閱CodeDeploy DeploymentGroup 《AWS CloudFormation 使用指南》 GitHubLocation中的 CodeDeploy DeploymentGroup 部署修訂版 S3Location 和部署修訂。

[2] 我們提供的範本可讓您在受支援的區域中建立 Amazon EC2 執行個體。 CodeDeploy 如需使用這些範本的詳細資訊，請參閱為 CodeDeploy (AWS CloudFormation 範本) 建立 Amazon EC2 執行個體。

[3] 此部署組態僅支援 Amazon ECS 藍色/綠色部署。如需 Amazon ECS 藍/綠部署的部署組態的詳細資訊 AWS CloudFormation，請參閱。AWS CloudFormation 藍/綠部署的部署組態 (Amazon ECS)如需 Amazon ECS 藍/綠部署的相關資訊，以 AWS CloudFormation 及如何在中檢視部署的詳細資訊 CodeDeploy，請參閱。透過以下方式建立亞馬遜ECS 藍色/綠色部署 AWS CloudFormation

搭 CodeDeploy 配 Amazon Virtual Private Cloud 使用

如果您使用 Amazon Virtual Private Cloud (Amazon VPC) 託管資 AWS 源，則可以在 VPC 和. 之間建立私有連接。 CodeDeploy您可以使用此連線 CodeDeploy 來啟用與 VPC 上的資源進行通訊，而無需透過公用網際網路。

Amazon VPC 是一項 AWS 服務，可用於在您定義的虛擬網路中啟動 AWS 資源。您可利用 VPC 來控制您的網路設定，例如 IP 地址範圍、子網路、路由表和網路閘道。使用 VPC 端點時，VPC 和 AWS 服務之間的路由由由 AWS 網路處理，您可以使用 IAM 政策來控制對服務資源的存取。

若要將 VPC 連接到 CodeDeploy，您需要為的定義介面 VPC 端點。 CodeDeploy介面端點是具有私有 IP 位址的 elastic network interface，可做為傳送至支援 AWS 服務之流量的進入點。端點提供可靠、可擴充的連線能力， CodeDeploy 無需網際網路閘道、網路位址轉譯 (NAT) 執行個體或 VPN 連線。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的什麼是 Amazon VPC。

介面 VPC 私人雲端端點的支援是一種使用具有私有 IP 位址的 elastic network interface AWS PrivateLink，在 AWS 服務之間進行私人通訊的 AWS 技術。如需詳細資訊，請參閱 AWS PrivateLink。

下列步驟適用於 Amazon VPC 的使用者。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的入門。

可用性

CodeDeploy 具有兩個 VPC 端點：一個用於 CodeDeploy 代理程式作業，另一個用於 CodeDeploy API 作業。下表顯示每個端點的支援 AWS 區域。

區域名稱	區域代碼	代理端點	API 端點
美國東部 (維吉尼亞北部)	us-east-1	是	是
美國東部 (俄亥俄)	us-east-2	是	是
美國西部 (加利佛尼亞北部)	us-west-1	是	是
美國西部 (奧勒岡)	us-west-2	是	是
非洲 (開普敦)	af-south-1	是	否
亞太區域 (香港)	ap-east-1	是	是
亞太區域 (海德拉巴)	ap-south-2	是	否
亞太區域 (雅加達)	ap-southeast-3	是	否
亞太區域 (墨爾本)	ap-southeast-4	是	否
亞太區域 (孟買)	ap-south-1	是	是

區域名稱	區域代碼	代理端點	API 端點
亞太區域 (大阪)	ap-northeast-3	是	否
亞太區域 (首爾)	ap-northeast-2	是	是
亞太區域 (新加坡)	ap-southeast-1	是	是
亞太區域 (雪梨)	ap-southeast-2	是	是
亞太區域 (東京)	ap-northeast-1	是	是
加拿大 (中部)	ca-central-1	是	是
中國 (北京)	cn-north-1	是	否
中國 (寧夏)	cn-northwest-1	否	否
歐洲 (法蘭克福)	eu-central-1	是	是
歐洲 (愛爾蘭)	eu-west-1	是	是
歐洲 (倫敦)	eu-west-2	是	是
歐洲 (米蘭)	eu-south-1	是	否
歐洲 (巴黎)	eu-west-3	是	是
歐洲 (西班牙)	eu-south-2	是	否
歐洲 (斯德哥爾摩)	eu-north-1	是	是
歐洲 (蘇黎世)	eu-central-2	是	否
以色列 (特拉維夫)	il-central-1	是	是
中東 (巴林)	me-south-1	是	是
中東 (阿拉伯聯合大公國)	me-central-1	是	否
南美洲 (聖保羅)	sa-east-1	是	是

區域名稱	區域代碼	代理端點	API 端點
AWS GovCloud (美國東部)	us-gov-east-1	否	否
AWS GovCloud (美國西部)	us-gov-west-1	否	否

為以下項目建立 VPC 端點 CodeDeploy

若要開 CodeDeploy 始使用您的 VPC，請為. CodeDeploy CodeDeploy代理程式 Git 作業和 CodeDeploy API 作業需要個別的端點。根據您的商業需求，您可能需要建立多個 VPC 端點。為建立 VPC 端點時 CodeDeploy，請選擇AWS 服務，然後在服務名稱中選擇下列選項：

- COM. 亞馬遜。 ## .codedeploy: 如果您要建立用 CodeDeploy 於 API 作業的虛擬私人雲端端點，請選擇此選項。例如，如果您的使用者使用 AWS CLI、 CodeDeploy API 或 AWS SDK 來與作業 (例如、和`ListDeploymentGroups`) 互 CodeDeploy 動 `CreateApplicationGetDeployment`，請選擇此選項。
- COM. 亞馬遜。 ##。 codedeploy-commands-secure：如果要為 CodeDeploy 代理程式作業建立 VPC 端點，請選擇此選項。您還需要在代理程式設`:enable_auth_policy:`定檔`true`中設定為，並附加必要的權限。如需詳細資訊，請參閱 設定代 CodeDeploy 理程式和 IAM 許可。

如果您使用的是 Lambda 或 ECS 部署，您只需要建立 VPC 端點即可。 ##. 代碼部署。使用 Amazon EC2 部署的客戶都需要使用 VPC 端點。 ##. 代碼部署和使用. ##。 codedeploy-commands-secure。

設定代 CodeDeploy 理程式和 IAM 許可

若要搭配使用 Amazon VPC 端點 CodeDeploy，您必須`true`在 EC2 或現場部署執行個體上的代理程式組態檔案中`:enable_auth_policy:`將值設定為。如需代理程式組態檔的詳細資訊，請參閱CodeDeploy 用戶端組態參考。

您還必須將以下 IAM 許可新增到 Amazon EC2 執行個體設定檔 (如果您使用 Amazon EC2 執行個體) 或 IAM 使用者或角色 (如果您使用的是現場部署執行個體)。

```
{
  "Statement": [
    {
```

```
      "Action": [
        "codedeploy-commands-secure:GetDeploymentSpecification",
        "codedeploy-commands-secure:PollHostCommand",
        "codedeploy-commands-secure:PutHostCommandAcknowledgement",
        "codedeploy-commands-secure:PutHostCommandComplete"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

如需詳細資訊，請參閱 Amazon VPC 使用者指南中的建立界面端點。

CodeDeploy 資源套件參考

許多 CodeDeploy 依賴的檔案都存放在公開可用的 AWS 區域特定 Amazon S3 儲存貯體中。這些檔案包括 CodeDeploy 代理程式、範本和範例應用程式檔案的安裝檔案。我們稱這個檔案集合為資 CodeDeploy 源工具組。

主題

- 依區域的資源套件時段名稱
- 資源套件內容
- 顯示資源套件檔案清單
- 下載資源套件檔案

依區域的資源套件時段名稱

此表格列出指南中某些程序必要的 *bucket-name* 取代的名稱。這些是包含 CodeDeploy 資源套件檔案的 Amazon S3 儲存貯體的名稱。

> **Note**
>
> 若要存取亞太區域 (香港) 區域的 Amazon S3 儲存貯體，您必須在 AWS 帳戶中啟用該區域。如需詳細資訊，請參閱管理 AWS 區域。

區域名稱	取#####	區域識別碼
美國東部 (維吉尼亞北部)	aws-codedeploy-us-east-1	us-east-1
美國東部 (俄亥俄)	aws-codedeploy-us-east-2	us-east-2
美國西部 (加利佛尼亞北部)	aws-codedeploy-us-west-1	us-west-1
美國西部 (奧勒岡)	aws-codedeploy-us-west-2	us-west-2
非洲 (開普敦)	aws-codedeploy-af-south-1	af-south-1
亞太區域 (香港)	aws-codedeploy-ap-east-1	ap-east-1
亞太區域 (海德拉巴)	aws-codedeploy-ap-south-2	ap-south-2
亞太區域 (雅加達)	aws-codedeploy-ap-southeast-3	ap-southeast-3
亞太區域 (墨爾本)	aws-codedeploy-ap-southeast-4	ap-southeast-4
亞太區域 (孟買)	aws-codedeploy-ap-south-1	ap-south-1
亞太區域 (大阪)	aws-codedeploy-ap-northeast-3	ap-northeast-3
亞太區域 (首爾)	aws-codedeploy-ap-northeast-2	ap-northeast-2
亞太區域 (新加坡)	aws-codedeploy-ap-southeast-1	ap-southeast-1
亞太區域 (悉尼)	aws-codedeploy-ap-southeast-2	ap-southeast-2
亞太區域 (東京)	aws-codedeploy-ap-northeast-1	ap-northeast-1
加拿大 (中部)	aws-codedeploy-ca-central-1	ca-central-1

區域名稱	取#####	區域識別碼
歐洲 (法蘭克福)	aws-codedeploy-eu-central-1	eu-central-1
歐洲 (愛爾蘭)	aws-codedeploy-eu-west-1	eu-west-1
歐洲 (倫敦)	aws-codedeploy-eu-west-2	eu-west-2
歐洲 (米蘭)	aws-codedeploy-eu-south-1	eu-south-1
Europe (Paris)	aws-codedeploy-eu-west-3	eu-west-3
歐洲 (西班牙)	aws-codedeploy-eu-south-2	eu-south-2
歐洲 (斯德哥爾摩)	aws-codedeploy-eu-north-1	eu-north-1
歐洲 (蘇黎世)	aws-codedeploy-eu-central-2	eu-central-2
以色列 (特拉維夫)	aws-codedeploy-il-central-1	il-central-1
Middle East (Bahrain)	aws-codedeploy-me-south-1	me-south-1
中東 (阿拉伯聯合大公國)	aws-codedeploy-me-central-1	me-central-1
南美洲 (聖保羅)	aws-codedeploy-sa-east-1	sa-east-1
AWS GovCloud (美國東部)	aws-codedeploy-us-gov-東方-1	us-gov-east-1
AWS GovCloud (美國西部)	aws-codedeploy-us-gov-西部-1	us-gov-west-1

資源套件內容

下表列出資 CodeDeploy 源套件中的檔案。

檔案	描述
LATEST_VERSION	Amazon EC2 系統管理員等更新機制所使用的檔案，用來判斷 CodeDeploy 代理程式的最新版本。
VERSION	在 CodeDeploy 代理程式版本 1.1.0 中移除了自動更新機制，不再使用此檔案。 CodeDeploy代理程式在執行個體上執行時自行更新的檔案。
codedeploy-agent.noarch.rpm	Amazon Linux 和紅帽企業 Linux (RHEL) 的 CodeDeploy 代理程式。可能有數個檔案具有相同的基礎檔案名稱，但有不一樣的版本 (例如 -1.0-0)。
codedeploy-agent_all.deb	Ubuntu 伺服器的 CodeDeploy 代理程式。可能有數個檔案具有相同的基礎檔案名稱，但有不一樣的版本 (例如 _1.0-0)。
codedeploy-agent.msi	適用於 Windows 伺服器的 CodeDeploy 代理程式。可能有數個檔案具有相同的基礎檔案名稱，但有不一樣的版本 (例如 -1.0-0)。
install	可用來更輕鬆地安裝 CodeDeploy 代理程式的檔案。
CodeDeploy_SampleCF_Template.json	您可以使用一到三個執行 Amazon Linux 或 Windows 伺服器的 Amazon EC2 執行個體啟動的 AWS CloudFormation 範本。可能有數個檔案具有相同的基礎檔案名稱，但有不一樣的版本 (例如 -1.0.0)。
CodeDeploy_SampleCF_ELB_Integration.json	可用來建立在 Apache Web 伺服器上執行之負載平衡範例網站的範本。 AWS CloudFormation 應用程式設定為跨您建立程式所在區域中的所有可用區域。此範本會建立三個 Amazon EC2 執行個體和 IAM 執行個體設定檔，以授與執行個體存取 Amazon S3、Amazon EC2 Auto

檔案	描述
	Scaling 和 Elastic Load Balancing 中的資源。AWS CloudFormation它也會建立負載平衡器和CodeDeploy服務角色。
SampleApp_ELB_Integration.zip	您可以部署到已註冊到 Elastic Load Balancing 器的 Amazon EC2 執行個體的範例應用程式修訂版。
SampleApp_Linux.zip	您可以部署到執行 Amazon Linux 的 Amazon EC2 執行個體或 Ubuntu 伺服器或 RHEL 執行個體的範例應用程式修訂版本。可能有數個檔案具有相同的基礎檔案名稱，但有不一樣的版本 (例如 -1.0)。
SampleApp_Windows.zip	您可以部署到 Windows 伺服器執行個體的範例應用程式修訂版本。可能有數個檔案具有相同的基礎檔案名稱，但有不一樣的版本 (例如 -1.0)。

顯示資源套件檔案清單

若要檢視檔案的清單，請使用您區域的 aws s3 ls 命令。

> **Note**
>
> 每個儲存貯體中的檔案，被設計來使用對應區域中的資源。

- `aws s3 ls --recursive s3://aws-codedeploy-us-east-2 --region us-east-2`
- `aws s3 ls --recursive s3://aws-codedeploy-us-east-1 --region us-east-1`
- `aws s3 ls --recursive s3://aws-codedeploy-us-west-1 --region us-west-1`
- `aws s3 ls --recursive s3://aws-codedeploy-us-west-2 --region us-west-2`

- `aws s3 ls --recursive s3://aws-codedeploy-ca-central-1 --region ca-central-1`
- `aws s3 ls --recursive s3://aws-codedeploy-eu-west-1 --region eu-west-1`
- `aws s3 ls --recursive s3://aws-codedeploy-eu-west-2 --region eu-west-2`
- `aws s3 ls --recursive s3://aws-codedeploy-eu-west-3 --region eu-west-3`
- `aws s3 ls --recursive s3://aws-codedeploy-eu-central-1 --region eu-central-1`
- `aws s3 ls --recursive s3://aws-codedeploy-il-central-1 --region il-central-1`
- `aws s3 ls --recursive s3://aws-codedeploy-ap-east-1 --region ap-east-1`
- `aws s3 ls --recursive s3://aws-codedeploy-ap-northeast-1 --region ap-northeast-1`
- `aws s3 ls --recursive s3://aws-codedeploy-ap-northeast-2 --region ap-northeast-2`
- `aws s3 ls --recursive s3://aws-codedeploy-ap-southeast-1 --region ap-southeast-1`
- `aws s3 ls --recursive s3://aws-codedeploy-ap-southeast-2 --region ap-southeast-2`
- `aws s3 ls --recursive s3://aws-codedeploy-ap-southeast-4 --region ap-southeast-4`
- `aws s3 ls --recursive s3://aws-codedeploy-ap-south-1 --region ap-south-1`
- `aws s3 ls --recursive s3://aws-codedeploy-sa-east-1 --region sa-east-1`

下載資源套件檔案

若要下載檔案，請使用 您區域的 aws s3 cp 命令。

> **Note**
>
> 請確定使用接近結束時間的期間 (.)。此會下載檔案到您目前的目錄。

例如，以下命令從其中一個儲存貯體的 /samples/latest/ 資料夾下載名為 SampleApp_Linux.zip 的單一檔案。

- ```
  aws s3 cp s3://aws-codedeploy-us-east-2/samples/latest/SampleApp_Linux.zip . --region
   us-east-2
  ```
- ```
  aws s3 cp s3://aws-codedeploy-us-east-1/samples/latest/SampleApp_Linux.zip . --region
   us-east-1
  ```
- ```
  aws s3 cp s3://aws-codedeploy-us-west-1/samples/latest/SampleApp_Linux.zip . --region
   us-west-1
  ```
- ```
  aws s3 cp s3://aws-codedeploy-us-west-2/samples/latest/SampleApp_Linux.zip . --region
   us-west-2
  ```
- ```
  aws s3 cp s3://aws-codedeploy-ca-central-1/samples/latest/SampleApp_Linux.zip . --
  region ca-central-1
  ```
- ```
  aws s3 cp s3://aws-codedeploy-eu-west-1/samples/latest/SampleApp_Linux.zip . --region
   eu-west-1
  ```
- ```
  aws s3 cp s3://aws-codedeploy-eu-west-2/samples/latest/SampleApp_Linux.zip . --region
   eu-west-2
  ```
- ```
  aws s3 cp s3://aws-codedeploy-eu-west-3/samples/latest/SampleApp_Linux.zip . --region
   eu-west-3
  ```
- ```
  aws s3 cp s3://aws-codedeploy-eu-central-1/samples/latest/SampleApp_Linux.zip . --
  region eu-central-1
  ```
- ```
  aws s3 cp s3://aws-codedeploy-il-central-1/samples/latest/SampleApp_Linux.zip . --
  region il-central-1
  ```
- ```
  aws s3 cp s3://aws-codedeploy-ap-east-1/samples/latest/SampleApp_Linux.zip . --region
   ap-east-1
  ```
- ```
  aws s3 cp s3://aws-codedeploy-ap-northeast-1/samples/latest/SampleApp_Linux.zip . --
  region ap-northeast-1
  ```

- ```
  aws s3 cp s3://aws-codedeploy-ap-northeast-2/samples/latest/SampleApp_Linux.zip . --region ap-northeast-2
  ```
- ```
  aws s3 cp s3://aws-codedeploy-ap-southeast-1/samples/latest/SampleApp_Linux.zip . --region ap-southeast-1
  ```
- ```
  aws s3 cp s3://aws-codedeploy-ap-southeast-2/samples/latest/SampleApp_Linux.zip . --region ap-southeast-2
  ```
- ```
  aws s3 cp s3://aws-codedeploy-ap-southeast-4/samples/latest/SampleApp_Linux.zip . --region ap-southeast-4
  ```
- ```
  aws s3 cp s3://aws-codedeploy-ap-south-1/samples/latest/SampleApp_Linux.zip . --region ap-south-1
  ```
- ```
  aws s3 cp s3://aws-codedeploy-sa-east-1/samples/latest/SampleApp_Linux.zip . --region sa-east-1
  ```

若要下載所有的檔案，請使用您區域的以下其中一個命令：

- ```
  aws s3 cp --recursive s3://aws-codedeploy-us-east-2 . --region us-east-2
  ```
- ```
  aws s3 cp --recursive s3://aws-codedeploy-us-east-1 . --region us-east-1
  ```
- ```
  aws s3 cp --recursive s3://aws-codedeploy-us-west-1 . --region us-west-1
  ```
- ```
  aws s3 cp --recursive s3://aws-codedeploy-us-west-2 . --region us-west-2
  ```
- ```
  aws s3 cp --recursive s3://aws-codedeploy-ca-central-1 . --region ca-central-1
  ```
- ```
  aws s3 cp --recursive s3://aws-codedeploy-eu-west-1 . --region eu-west-1
  ```
- ```
  aws s3 cp --recursive s3://aws-codedeploy-eu-west-2 . --region eu-west-2
  ```
- ```
  aws s3 cp --recursive s3://aws-codedeploy-eu-west-3 . --region eu-west-3
  ```
- ```
  aws s3 cp --recursive s3://aws-codedeploy-eu-central-1 . --region eu-central-1
  ```
  ```

- `aws s3 cp --recursive s3://aws-codedeploy-il-central-1 . --region il-central-1`
- `aws s3 cp --recursive s3://aws-codedeploy-ap-east-1 . --region ap-east-1`
- `aws s3 cp --recursive s3://aws-codedeploy-ap-northeast-1 . --region ap-northeast-1`
- `aws s3 cp --recursive s3://aws-codedeploy-ap-northeast-2 . --region ap-northeast-2`
- `aws s3 cp --recursive s3://aws-codedeploy-ap-southeast-1 . --region ap-southeast-1`
- `aws s3 cp --recursive s3://aws-codedeploy-ap-southeast-2 . --region ap-southeast-2`
- `aws s3 cp --recursive s3://aws-codedeploy-ap-southeast-4 . --region ap-southeast-4`
- `aws s3 cp --recursive s3://aws-codedeploy-ap-south-1 . --region ap-south-1`
- `aws s3 cp --recursive s3://aws-codedeploy-sa-east-1 . --region sa-east-1`

CodeDeploy 配額

下表說明中的配額 CodeDeploy。

> **Note**
>
> EC2 /內部部署在小時內部署限制不同而異。對於 2023 年 6 月之前建立的自訂部署組態，限制為 8 小時。對於 2023 年 6 月或更新版本建立的自訂部署組態，限制為 12 小時。對於預先定義的部署組態，限制為 12 小時。

名稱	預設	可調整	描述
AWS 部 Lambda 可在數小時內執行	每個受支援的區域：50	否	AWS Lambda 部署可執行的最大小時數 (第一次和最後一次流量轉移之間的

名稱	預設	可調整	描述
			最長時間為 48 小時，加上兩個可能的生命週期勾點中的每一個小時)
每個區域每個帳戶關聯的應用	每個受支援的區域：1,000	是	單一地區中與 AWS 帳戶相關聯的應用程式數目上限
每個部署群組的關聯警示	每個受支援的區域：20	是	與部署群組相關聯的警示數目上限
部署群組中的自動調整資源分組	每個受支援的區域：10	是	部署群組中 Amazon EC2 Auto Scaling 群組的數目上限
每個帳戶的並行部署	每個受支援的區域：1,000	是	與 AWS 帳戶相關聯的同時部署數目上限。Amazon EC2 Auto Scaling 群組中向上擴充的 Amazon EC2 執行個體的每個部署都會計為 EC2 執行個體關聯的每個應用程式的單一並行部署。
每個部署群組的同時部署	每個受支援的區域：1	否	同時部署至部署群組的最大數目。此限制可防止同時將相同應用程式部署至相同的部署群組。
每個帳戶的自訂部署組態	每個受支援的區域：50	否	與 AWS 帳戶相關聯的自訂部署組態數目上限
與單一應用程式相關聯的部署群組	每個受支援的區域：1,000	是	與單一應用程式相關聯的部署群組上限數量

名稱	預設	可調整	描述
EC2/內部部署藍/綠部署在數小時內執行	每個支持地區：109	否	EC2 /內部部署藍/綠部署可以執行的最大小時數 (上述兩個限制中的每個限制為 48 小時，加上 13 個可能的生命週期事件中每個一個小時)
EC2/內部部署在數小時內執行	每個支持地區：12	否	EC2/內部部署可以執行的最大小時間
部署群組中的事件通知觸發器	每個受支援的區域：10	是	部署群組中事件通知觸發的上限數量
GitHub 每個帳戶的連接令牌	每個受支援的區域：25	否	單個 AWS 帳戶的最大 GitHub 連接令牌數
在 EC2 /內部部署藍/綠部署期間部署完成到原始執行個體終止之間的時間	每個支持地區：48	否	在 EC2 /內部部署藍/綠部署期間部署完成到原始執行個體終止之間的最大小時間
在 EC2 /內部部署藍/綠部署期間的修訂版部署到流量轉移到替換執行個體的間隔時間	每個支持地區：48	否	在 EC2 /內部部署藍/綠部署期間的修訂版部署與流量轉移到替換執行個體之間的最大小時間
每部署執行個體計數	us-east-1：2 千 每個其他支援的區域：1,000 個	是	在單一部署中的執行個體上限數量
成功部署後，藍/綠部署可以等待的分鐘數，然後才能從原始部署終止執行個體	每個支持的地區：2,800	否	成功部署後，從原始部署終結執行個體前，藍/綠部署可以等待的最久分鐘數

名稱	預設	可調整	描述
AWS Lambda 初期測試或線性部署期間的第一個和最後一個流量變化之間的分	每個支持的地區：2,880	否	AWS Lambda 初期測試或線性部署期間，第一個和最後一個流量轉移之間的最大分鐘數
如果生命週期事件未啟動，直到部署失敗前的分鐘	每個受支援的區域：5	否	如果生命週期事件在 (1) 使用主控台或 AWS CLI create-deployment 命令觸發部署，或 (2) 先前的生命週期事件已完成，則部署失敗前的分鐘數上限。
可與 Amazon ECS 服務關聯的部署群組數目	每個受支援的區域：1	否	可與 Amazon ECS 服務關聯的部署群組數目上限
可傳遞至 BatchGetOnPremises Instances API 動作的執行個體數目	每個受支援的區域：100	否	可傳遞至 BatchGetOnPremisesInstances API 動作的執行個體數目上限
每個帳戶正在進行的並行部署所使用的執行個體數目	us-east-1：3 千 每個其他支援的區域：1,000 個	是	可使用於進行中且和帳戶有關的同步部署之執行個體最大數量
Amazon ECS 部署期間流量路由的接聽程式數目	每個受支援的區域：1	否	在 Amazon ECS 部署期間，流量路由的接聽程式數目上限
如果未完成，則部署生命週期事件失敗前的秒數	每個支持的地區：3,600 秒	否	如果尚未完成，部署生命週期事件失敗前可持續的最久秒數
部署群組名稱的大小	每個受支援的區域：100	否	部署群組名稱的最大字元數

名稱	預設	可調整	描述
標籤鍵的大小	每個支持地區：128	否	標籤金鑰的最大字元數。
標籤值的大小	每個支持的地區：256	否	標籤數值的最大字元數
部署群組中的標記	每個受支援的區域：10	否	部署群組可有的最大標籤數
AWS Lambda 部署期間可以以一個遞增方式移動的流量	每個支持地區：99	否	AWS Lambda 部署期間可以一次增量移動的流量的最大百分比

疑難排 CodeDeploy

使用本節中的主題可協助您解決使用時可能遇到的問題和錯誤 CodeDeploy。

> **Note**
>
> 您可以藉由檢閱在部署程序期間建立的日誌檔，識別許多部署故障的原因。為了簡單起見，我們建議您使用 Amazon CloudWatch Logs 集中監控日誌檔案，而不是逐個檢視日誌檔案。如需相關資訊，請參閱Monitoring Deployments with Amazon CloudWatch Tools。

主題

一般故障診斷問題

主題

一般故障診斷檢查清單

您可以使用以下檢查清單來排除故障的部署。

1. 請參閱 檢視 CodeDeploy 部署詳情 和 View Instance Details 以判斷部署失敗的原因。如果您無法判斷原因，請檢閱此檢查清單中的項目。
2. 請檢查執行個體的設定是否正確：
   - 執行個體是否使用指定 EC2 key pair 啟動？ 如需詳細資訊，請參閱 Amazon EC2 使用者指南中的 EC2 金鑰配對。
   - 是否將正確的 IAM 執行個體設定檔連接到執行個體？ 如需詳細資訊，請參閱 設定要使用的 Amazon EC2 執行個體 CodeDeploy 及 步驟 4：為您的 Amazon EC2 執行個體建立 IAM 執行個體設定檔。
   - 該執行個體是否有加上標籤？ 如需詳細資訊，請參閱 Amazon EC2 使用者指南中的主控台中的使用標籤。
   - CodeDeploy 代理程式是否已在執行個體上安裝、更新和執行？ 如需詳細資訊，請參閱 管理 CodeDeploy 代理程式作。若要檢查已安裝的代理程式版本，請參閱判斷代 CodeDeploy理程式的版本。
3. 檢查應用程式和部署群組設定：
   - 若要查看您的應用程式設定的詳細資訊，請參閱 檢視申請詳細資料 CodeDeploy。
   - 若要查看您的部署群組設定的詳細資訊，請參閱檢視部署群組詳細資料 CodeDeploy。
4. 確認是否已正確設定應用程式修訂版：
   - 檢查 AppSpec 檔案的格式。如需詳細資訊，請參閱 將應用程式規格檔案新增至修訂 CodeDeploy 及 CodeDeploy AppSpec 檔案參考。
   - 檢查您的 Amazon S3 儲存貯體或 GitHub 儲存庫，確認應用程式修訂版本位於預期的位置。
   - 檢閱 CodeDeploy 應用程式修訂版的詳細資訊，以確定已正確註冊。如需相關資訊，請參閱使用檢視應用程式修訂版 CodeDeploy。
   - 如果您是從 Amazon S3 進行部署，請檢查 Amazon S3 儲存貯體以確認是否 CodeDeploy 已獲得下載應用程式修訂版本的許可。如需儲存貯體政策的資訊，請參閱 部署先決條。
   - 如果您要從部署 GitHub，請檢查您的 GitHub 存儲庫以驗證是否 CodeDeploy已授予下載應用程序修訂版的權限。如需詳細資訊，請參閱 使用建立部署 CodeDeploy 及 GitHub 使用應用程式的驗證 CodeDeploy。
5. 檢查服務角色的設定是否正確。如需相關資訊，請參閱步驟 2：建立服務角色 CodeDeploy。
6. 確認您依照中 開始使用 CodeDeploy 的步驟執行：
   - 提供具有適當權限的使用者。

- 選擇安裝或升級，並設定 AWS CLI。
- 建立 IAM 執行個體設定檔和服務角色。

如需詳細資訊，請參閱 適用於 AWS CodeDeploy的 Identity and Access Management。

7. 確認您使用的是 1.6.1 或更新 AWS CLI 版本。若要檢查安裝的版本，請呼叫 aws --version。

如果您仍然無法排除故障的部署，請檢閱本主題中的其他問題。

CodeDeploy 僅部分 AWS 區域支援部署資源

如果您看不到或無法從或 CodeDeploy 主控台存取應用程式、部署群組、執行個體或其他部署資源，請確定您參考的是 [Region] 和 [中的端點] 中列出的其中一個區 AWS 域AWS 一般參考。 AWS CLI

CodeDeploy 部署中使用的 EC2 執行個體和 Amazon EC2 Auto Scaling 群組必須在其中一個 AWS 區域中啟動並建立。

如果您使用的是 AWS CLI，請從執行aws configure命令 AWS CLI。然後，您可以查看和設置默認 AWS 區域。

如果您使用 CodeDeploy 主控台，請在導覽列上，從區域選取器中選擇其中一個支援的區 AWS 域。

> **Important**
>
> 若要在中國 (北京) 區域或中國 (寧夏) 區域使用服務，您必須擁有這些地區的帳戶和憑證。其他 AWS 地區的帳戶和憑證不適用於北京和寧夏地區，反之亦然。
> 有關中國區域某些資源的資訊，例如 CodeDeploy Resource Kit 值區名稱和 CodeDeploy 代理程式安裝程序，不包含在此版本的CodeDeploy 使用者指南中。
> 如需詳細資訊：
>
> - CodeDeployin 中國 (北京) 地區開始使用 AWS
> - CodeDeploy 中國地區用戶指南 (英文版本 | 中文版)

本指南中的程序與 CodeDeploy 主控台不相符

本指南中的程序反映新的主控台設計。如果您正在使用較舊版本的主控台，本指南中的許多概念和基本程序仍適用 。若要存取新主控台的說明，請選擇資訊圖示。

無法使用必要的 IAM 角色

如果您依賴 IAM 執行個體設定檔或建立為 AWS CloudFormation 堆疊一部分的服務角色，如果刪除堆疊，也會刪除所有 IAM 角色。這可能是 IAM 角色不再顯示在 IAM 主控台中的原因，而且 CodeDeploy 不再如預期般運作的原因。若要修正此問題，您必須手動重新建立已刪除的 IAM 角色。

使用某些文字編輯器建立 AppSpec檔案和 shell 指令碼可能會導致部署失敗

有些文字編輯器將引進不符合、非列印字元到檔案裡。如果您使用文字編輯器建立或修改 AppSpec 要在 Amazon Linux、Ubuntu 伺服器或 RHEL 執行個體上執行的檔案或殼層指令碼檔案，則任何依賴這些檔案的部署都可能會失敗。在部署期間 CodeDeploy 使用這些檔案時，如果存在這些字元，可能會導致 hard-to-troubleshoot AppSpec 檔案驗證失敗和指令碼執行失敗。

在 CodeDeploy 主控台的部署事件詳細資料頁面上，選擇 [檢視記錄]。（或者您可 AWS CLI 以使用調用get-deployment-instance命令。）尋找錯誤，例如：`invalid character`、`command not found`，或 `file not found`。

若要解決此問題，我們有以下建議：

- 請勿使用將非列印字元 (例如換行字元) 引入 AppSpec 檔案和 shell 指令碼檔案的文`^M`字編輯器。
- 使用顯示非列印字元的文字編輯器，例如 AppSpec 檔案中的換行字元和 shell 指令碼檔案，以便您可以尋找並移除任何可能引入的字元。如需這些類型文字編輯器的範例，請至網際網路搜尋顯示換行符號的文字編輯器。
- 使用在 Amazon Linux、Ubuntu 伺服器或 RHEL 執行個體上執行的文字編輯器，建立在 Amazon Linux、Ubuntu 伺服器或 RHEL 執行個體上執行的殼層指令碼檔案。如需這些類型文字編輯器的範例，請至網際網路搜尋「Linux shell 指令碼編輯器」。
- 如果您必須使用 Windows 或 macOS 中的文字編輯器來建立要在 Amazon Linux、Ubuntu 伺服器或 RHEL 執行個體上執行的殼層指令碼檔案，請使用將 Windows 或 macOS 格式文字轉換為 Unix 格式的程式或公用程式。如需這些程式和公用程式的範例，請至網際網路搜尋「DOS 轉換 UNIX」或「Mac 轉換 UNIX」。請務必在目標作業系統上測試轉換過的 shell 指令碼檔案。

使用 macOS 的 Finder 套用應用程式修訂可能導致失敗

如果您在 Mac 上使用 Finder 圖形使用者介面 (GUI) 應用程式，將 AppSpec 檔案及相關檔案和指令碼捆綁到應用程式修訂封存檔 (.zip) 檔案中，部署可能會失敗。這是因為 Finder 會在 .zip 檔案中建立中繼`__MACOSX`資料夾，並將元件檔案放入其中。 CodeDeploy 找不到元件檔案，因此部署失敗。

若要解決這個問題，建議您使用呼叫 push 命令，該命令會將元件檔案壓縮到預期的結構中。 AWS CLI 或者，您可以使用 GUI 的終端機壓縮元件檔案。終端機不會建立中繼 __MACOSX 資料夾。

疑難排解 EC2/內部部署問題

主題

> **Note**
>
> 您可以藉由檢閱在部署程序期間建立的日誌檔，識別許多部署故障的原因。為了簡單起見，我們建議您使用 Amazon CloudWatch Logs 集中監控日誌檔案，而不是逐個檢視日誌檔案。如需詳細資訊，請參閱在 CodeDeploy 記 CloudWatch 錄主控台中檢視記錄

> **Tip**
>
> 如需自動執行許多與 EC2 /內部部署相關的故障排解任務的 Runbook，請參閱 AWS Systems Manager 自動化手冊參考資料TroubleshootCodeDeploy中的 AWSSupport-。

CodeDeploy 插件 CommandPoller缺少憑據錯誤

如果您收到與 `InstanceAgent::Plugins::CodeDeployPlugin::CommandPoller: Missing credentials - please check if this instance was started with an IAM instance profile` 類似的錯誤，可能是由以下其中一項造成：

- 您要部署的執行個體沒有與其相關聯的 IAM 執行個體設定檔。
- 您的 IAM 執行個體設定檔未設定正確的許可。

IAM 執行個體設定檔授與 CodeDeploy 代理程式通訊 CodeDeploy和從 Amazon S3 下載修訂版的權限。若為 EC2 執行個體，請參閱適用於 AWS CodeDeploy的 Identity and Access Management。如需現場部署執行個體的詳細資訊，請參閱 Working with On-Premises Instances。

部署失敗訊息：「PKCS7 簽章訊息驗證失敗」

此錯誤訊息指出執行個體正在執行僅支援 SHA-1 雜湊演算法的 CodeDeploy 代理程式版本。SHA-2 雜湊演算法的 Support 已於 2015 年 11 月發行的 CodeDeploy 代理程式版本 1.0.1.854 中引入。自 2016 年 10 月 17 日起，如果安裝的 CodeDeploy 代理程式版本早於 1.0.1.854，則部署會失敗。如需詳細資訊，請參閱AWS 若要切換至 SSL 憑證的 SHA256 雜湊演算法，請注意：淘汰舊版本 1.0.1.85 的 CodeDeploy 主機代理程式和。更新 CodeDeploy 代理程式

部署或重新部署相同的檔案到同一個執行個體裡時，發生錯誤訊息「部署失敗，因為指定的檔案已存在於此位置」

當 CodeDeploy 嘗試將檔案部署到執行個體，但指定的目標位置中已存在相同名稱的檔案時，該執行個體的部署可能會失敗。您可能會收到「部署失敗，因為指定的檔案已存在於此位置：*location-name*」的錯誤訊息。這是因為，在每次部署期間， CodeDeploy 首先會刪除先前部署中列示在清理記錄檔中的所有檔案。如果目標安裝資料夾中有未列在此清除檔案中的檔案，則 CodeDeploy 代理程式預設會將此解譯為錯誤，且部署失敗。

> **Note**
>
> 在 Amazon Linux、RHEL 和 Ubuntu 伺服器執行個體上，清理檔案位於`/opt/codedeploy-agent/deployment-root/deployment-instructions/`. 在 Windows 伺服器執行個體上，位置為`C:\ProgramData\Amazon\CodeDeploy\deployment-instructions\`。

最簡單防止此錯誤發生的方式是指定選項，以避免預設行為使部署失敗。對於每個部署，您可以選擇是否使部署失敗、是否覆寫這些未列在清除檔案中的檔案，或是否保留已存在於執行個體中的檔案。

覆寫選項功能很實用，例如：在最後一次部署後，您手動將檔案放置到執行個體上，然後新增相同名稱的檔案到下一個應用程式修定版中。

您可以選擇為放置到執行個體的檔案選擇保留選項，選擇您下一個部署想要含有的檔案，此舉一來則可以不用將這些檔案加入到應用程式修訂套件中。如果您的應用程式檔案已存在 CodeDeploy 於生產環境中，並且您想要第一次使用進行部署，則保留選項也很有用。如需詳細資訊，請參閱 建立 EC2/內部部署計算平台部署 (主控台) 及 復原現有內容的行為。

對 `The deployment failed because a specified file already exists at this location` 部署問題進行故障診斷

如果您選擇不指定覆寫或保留在目標部署位置 CodeDeploy偵測到的內容的選項 (或未指定任何部署選項來處理程式化命令中的現有內容)，則可以選擇對錯誤進行疑難排解。

以下資訊僅適用於當您選擇不保留或覆寫內容。

如果您嘗試重新部署具有相同名稱和位置的檔案，則如果您使用之前使用的相同基礎部署群組識別碼指定應用程式名稱和部署群組，則重新部署的可能性較高。 CodeDeploy 使用基礎部署群組 ID 識別要在重新部署之前移除的檔案。

部署新的檔案或重新部署相同的檔案到執行個體上，其失敗的可能原因：

- 重新部署同個修訂時，您指定不同的應用程式名稱到同一個執行個體上。重新部署失敗，因為即使部署群組名稱是相同的，但使用不同的應用程式名稱表示使用不同的基礎部署群組 ID。
- 您刪除了應用程式的部署群組後並為其重建，接著嘗試將相同修訂重新部署至該部署群組。重新部署會失敗，因為即使部署群組名稱相同，也會 CodeDeploy 參考不同的基礎部署群組識別碼。
- 您在中刪除了應用程式和部署群組 CodeDeploy，然後使用與刪除的名稱相同的新應用程式和部署群組建立。然後，您嘗試重新部署之前部署到部署群組的修訂到新建立且有相同名稱的部署群組裡。重新部署會失敗，因為即使應用程式和部署群組名稱相同， CodeDeploy 仍會參考您刪除之部署群組的識別碼。
- 您部署修訂到部署群組裡，然後部署相同的修訂至同一個執行個體裡的另一個部署群組。第二個部署失敗，因為 CodeDeploy 參考了不同的基礎部署群組識別碼。
- 您部署修訂到一個部署群組裡，然後部署另一個的修訂至同一個執行個體裡的另一個部署群組。在相同位置中，至少有一個相同名稱的檔案，在此位置中，第二部署群組會嘗試部署。第二個部署失敗，因為 CodeDeploy 在第二次部署開始之前不會移除現有檔案。兩種部署 > 參考不同的部署群組 ID。

- 您已在中部署了修訂版 CodeDeploy，但至少有一個名稱相同且位於相同位置的檔案。部署失敗，因為依預設， CodeDeploy 不會在部署開始之前移除現有檔案。

若要解決這些情況，請執行以下其中一項：

- 將檔案從先前部署的位置和執行個體中移除，然後再部署一次。
- 在修訂的 AppSpec 檔案中，在 ApplicationStop 或 BeforeInstall部署生命週期事件中，指定一個自訂腳本，以刪除與您的修訂即將安裝之檔案相符的任何位置的檔案。
- 部署或重新部署檔案到先前並不屬於部署的位置或執行個體上。
- 刪除應用程式或部署群組之前，請部署包含指定沒有 AppSpec 檔案要複製到執行個體的檔案的修訂版。對於部署，指定應用程式名稱和部署群組名稱，其使用相同的基本應用程式和您將要刪除的部署群組相同的 ID。(您可以使用get-deployment-group命令擷取部署群組 ID。) CodeDeploy使用基礎部署群組 ID 和 AppSpec 檔案來移除先前成功部署中安裝的所有檔案。

長文件路徑導致「沒有這樣的文件或目錄」錯誤

對於 Windows 執行個體的部署，如果您的 appspec.yml 檔案的檔案區段中的檔案路徑大於 260 個字元，您可能會看到部署失敗，並顯示類似下列內容的錯誤：

```
No such file or directory @ dir_s_mkdir - C:\your-long-file-path
```

發生此錯誤的原因是 Windows 預設不允許檔案路徑大於 260 個字元，如微軟的文件所述。

如果是 CodeDeploy 代理程式版本 1.4.0 或更新版本，您可以根據代理程式安裝程序，以兩種方式啟用長檔案路徑：

如果尚未安裝 CodeDeploy 代理程式：

1. 在您計劃安裝 CodeDeploy 代理程式的電腦上，使用此命令啟用 `LongPathsEnabled` Windows 登錄機碼：

   ```
   New-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\FileSystem"
            -Name "LongPathsEnabled" -Value 1 -PropertyType DWORD -Force
   ```

2. 安裝代 CodeDeploy 理程式。如需詳細資訊，請參閱 安裝 CodeDeploy 代理程式。

如果已安裝 CodeDeploy 代理程式：

1. 在 CodeDeploy 代理程式電腦上，使用此命令啟用 LongPathsEnabled Windows 登錄機碼：

```
New-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\FileSystem"
-Name "LongPathsEnabled" -Value 1 -PropertyType DWORD -Force
```

2. 重新啟動 CodeDeploy 代理程式，讓登錄機碼變更生效。如果要重新啟動代理程式，請使用此命令：

```
powershell.exe -Command Restart-Service -Name codedeployagent
```

長時間執行的程序可能導致部署失敗

對於 Amazon Linux、Ubuntu 伺服器和 RHEL 執行個體的部署，如果您有啟動長時間執行程序的部署指令碼， CodeDeploy 可能會在部署生命週期事件中花費很長時間等待，然後部署失敗。這是因為如果處理序執行的時間長於前景處理序，而在該事件中預期的背景處理序需要執行、 CodeDeploy 停止和失敗部署，即使處理序仍如預期般執行也是如此。

例如，應用程式修訂在其根目錄下包含兩個檔案：after-install.sh 和 sleep.sh。其 AppSpec 檔案包含下列指示：

```
version: 0.0
os: linux
files:
  - source: ./sleep.sh
    destination: /tmp
hooks:
  AfterInstall:
    - location: after-install.sh
      timeout: 60
```

該after-install.sh文件在 AfterInstall 應用程序生命週期事件期間運行。以下是其內容：

```
#!/bin/bash
/tmp/sleep.sh
```

sleep.sh 檔案包含下列內容，會使程式暫停執行三分鐘 (180 秒)，模擬某些長時間執行的程序：

```
#!/bin/bash
sleep 180
```

當通after-install.sh話時sleep.sh，sleep.sh開始並執行三分鐘 (180 秒)，這是兩分鐘 (120 秒) 過去 CodeDeploy 預期的時間 sleep.sh (並且，根據關係after-install.sh) 停止執行。在一分鐘 (60 秒) 逾時之後，即使繼續如預期般執行，sleep.sh仍會在 AfterInstall 應用程式生命週期事件中 CodeDeploy 停止並失敗部署。隨即會顯示下列錯誤：

```
Script at specified location: after-install.sh failed to complete in 60 seconds.
```

僅在 after-install.sh 中新增 & 符號，無法在背景執行 sleep.sh。

```
#!/bin/bash
# Do not do this.
/tmp/sleep.sh &
```

這樣做可能會讓部署保持擱置狀態，最長可達預設的 1 小時部署生命週期事件逾時期間，之後會像以前一樣在 AfterInstall 應用程式生命週期事件中 CodeDeploy 停止並失敗部署。

在中after-install.sh，呼叫sleep.sh如下，可 CodeDeploy 在程序開始執行之後繼續執行：

```
#!/bin/bash
/tmp/sleep.sh > /dev/null 2> /dev/null < /dev/null &
```

在之前的呼叫中，sleep.sh 是您希望在背景開始執行的程序名稱，該程序會將 stdout、stderr 和 stdin 重新導向至 /dev/null。

疑難排解失敗的 AllowTraffic 生命週期事件，但部署記錄中未報告任何錯誤

在某些情況下，藍/綠部署會在 AllowTraffic 生命週期事件期間失敗，但部署記錄檔不會指出失敗的原因。

此失敗通常是因為 Classic Load Balancer、應用程式負載平衡器或 Network Load Balancer 用於管理部署群組流量的 Elastic Load Balancing 中設定健康狀態檢查錯誤所致。

若要解決該問題，請檢閱和修正負載平衡器的運作狀況檢查裡任何的錯誤。

對於傳統負載平衡器，請參閱傳統負載平衡器使用者指南和 Elastic Load Balancing API 參考版本 2012-06-01 ConfigureHealthCheck中的設定 Health 狀態檢查。

對於應用程式負載平衡器，請參閱應用程式負載平衡器使用者指南中的目標群組 Health 狀態檢查。

對於 Network Load Balancer，請參閱網路負載平衡器使用者指南中的目標群組的 Health 全狀況檢查。

疑難排解失敗 ApplicationStop BeforeBlockTraffic、或 AfterBlockTraffic 部署生命週期事件

在部署期間，CodeDeploy 代理程式會執行先前成功部署的 ApplicationStop BeforeBlockTraffic、和 AppSpec 檔案 AfterBlockTraffic 中指定的指令碼。(所有其他指令碼均從目前部署中的 AppSpec 檔案執行。) 如果這些指令碼中的其中一個包含錯誤且不能執行成功，則部署會失敗。

失敗的可能原因有：

- CodeDeploy 代理程式會在正確的位置尋找*deployment-group-id*_last_successful_install檔案，但*deployment-group-id*_last_successful_install檔案中列出的位置不存在。

  在 Amazon Linux、Ubuntu 伺服器和 RHEL 執行個體上，此檔案必須存在/opt/codedeploy-agent/deployment-root/deployment-instructions於.

  在 Windows 伺服器執行個體上，此檔案必須儲存在資C:\ProgramData\Amazon\CodeDeploy\deployment-instructions料夾中。
- 在檔案中列出的位置中，*deployment-group-id*_last_successful_install AppSpec 檔案無效或指令碼未成功執行。
- 此指令碼內含無法更正的錯誤，所以永遠不會順利執行。

使用主 CodeDeploy 控台來調查為什麼在任何這些事件期間部署可能會失敗。在部署的詳細資訊頁面上，選擇 View events (檢閱事件)。在執行個體的詳細資訊頁面上，在ApplicationStop、或資AfterBlockTraffic料列中 BeforeBlockTraffic，選擇 [檢視記錄檔]。或者使用 AWS CLI 來呼叫指get-deployment-instance令。

如果失敗的原因是上次成功部署中永遠不會成功執行的指令碼，請建立部署並指定應忽略 ApplicationStop BeforeBlockTraffic、和 AfterBlockTraffic 失敗。有兩種方式可以進行：

- 使用 CodeDeploy 主控台建立部署。在 [建立部署] 頁面的 [ApplicationStop 生命週期事件失敗] 下，選擇 [如果執行個體上的這個生命週期事件失敗，則不要讓部署失敗至執行個體]。
- 使 AWS CLI 用呼叫指create-deployment令並包含選--ignore-application-stop-failures項。

當您再次部署應用程式修訂，此部署會持續，即使這三個生命週期事件中的任何一個故障。如果新的修訂版包含這些生命週期事件的修正指令碼，則未來部署可在不套用此修正而成功執行。

疑難排解失敗的 DownloadBundle 部署生命週期事件 UnknownError：未開啟以供讀取

如果您嘗試從 Amazon S3 部署應用程式修訂版，且部署在部 DownloadBundle 署生命週期事件期間失敗，並顯示`UnknownError: not opened for reading`錯誤：

- 有內部 Amazon S3 服務錯誤。請再次部署應用程式修訂。
- EC2 執行個體上的 IAM 執行個體設定檔沒有存取 Amazon S3 中應用程式修訂版的許可。如需 Amazon S3 儲存貯體政策的相關資訊，請參閱將修訂推送 CodeDeploy 至 Amazon S3 (僅適用於 EC2 /內部部署)和部署先決條。
- 您部署到的執行個體與一個 AWS 區域 (例如美國西部 (奧勒岡)) 相關聯，但包含應用程式修訂版的 Amazon S3 儲存貯體與另一個 AWS 區域 (例如美國東部 (維吉尼亞北部)) 相關聯。確保應用程式修訂版位於與執行個體相關聯的相同 AWS 區域的 Amazon S3 儲存貯體中。

在部署的事件詳細資訊頁面，於 Download bundle (下載套用) 列，選取 View logs (檢視日誌)。或者使用 AWS CLI 來呼叫指get-deployment-instance令。如果發生此錯誤，輸出中應出現錯誤碼為 `UnknownError` 和錯誤訊息為 `not opened for reading` 的錯誤。

判斷此錯誤的原因：

1. 在至少一個執行個體上啟用有線登入，然後再部署應用程式修訂一次。
2. 檢查有線日誌檔尋找錯誤。此問題的常見錯誤訊息包含「拒絕存取」。
3. 檢查完日誌檔案後，我們建議您停用線路日誌記錄，以減少日誌檔案大小以及未來在執行個體上以純文字形式顯示在輸出中的敏感資訊量。

若要取得有關如何尋找配線記錄檔案以及啟用和停用電線記錄的資訊，請參閱CodeDeploy 代理程式組態參考`:log_aws_wire:`中的。

對所有生命週期事件略過錯誤進行故障診斷

如果跳過 EC2 或現場部署的所有生命週期事件，您可能會收到類似於`The overall deployment failed because too many individual instances failed deployment, too few healthy instances are available for deployment, or some instances in your deployment group are experiencing problems. (Error code: HEALTH_CONSTRAINTS)`. 以下是一些可能的原因和解決方案：

- CodeDeploy 代理程式可能未在執行個體上安裝或執行。如果要判斷 CodeDeploy 代理程式是否正在執行：
  - 對於 Amazon Linux RHEL 或 Ubuntu 伺服器，請執行下列動作：

    ```
    systemctl status codedeploy-agent
    ```

  - 對於 Windows，請執行下列動作：

    ```
    powershell.exe -Command Get-Service -Name CodeDeployagent
    ```

  如果未安裝或執行 CodeDeploy 代理程式，請參閱確認 CodeDeploy 代理程式正在執行。

  您的執行個體可能無法使用連接埠 443 到達 CodeDeploy 或 Amazon S3 公有端點。請嘗試執行下列其中一項操作：
  - 將公有 IP 地址指派給執行個體，並使用其路由表允許網際網路存取。請確定與執行個體關聯的安全群組允許透過連接埠 443 (HTTPS) 進行對外存取。如需詳細資訊，請參閱 CodeDeploy 代理程式的通訊協定和連接埠。
  - 如果執行個體是在私有子網路中佈建的，則使用 NAT 閘道，而非路由表中的網際網路閘道。如需更多詳細資訊，請參閱 NAT 閘道。
- 的服務角色 CodeDeploy 可能沒有必要的權限。如要設定 CodeDeploy 的服務角色，請參閱 步驟 2：建立服務角色 CodeDeploy
- 如果您使用 HTTP Proxy，請確定已在代 CodeDeploy 理程式組態檔的`:proxy_uri:`設定中指定該代理伺服器。如需詳細資訊，請參閱 CodeDeploy 用戶端組態參考。
- 您部署執行個體的日期和時間簽章，可能不符合您部署請求的日期和時間簽章。在 CodeDeploy 代理程式記錄檔`Cannot reach InstanceService: Aws::CodeDeployCommand::Errors::InvalidSignatureException - Signature expired`中尋找類似的錯誤。如果您看到此錯誤，請遵循以下步驟 疑難排解「InvalidSignatureException — 簽章已過期:[時間] 現在早於 [時間]」部署錯誤。如需詳細資訊，請參閱 檢視 CodeDeploy EC2/內部部署的記錄資料。
- CodeDeploy 代理程式可能會因為執行個體的記憶體或硬碟空間不足而停止執行。嘗試更新 CodeDeploy 代理程式組態中的`max_revisions`設定，以降低執行個體上的封存部署數目。如果您針對 EC2 執行個體執行此動作，但問題仍然存在，請考慮使用較大的執行個體。例如，如果您的執行個體類型是 `t2.small`，請嘗試使用 `t2.medium`。如需詳細資訊，請參閱 CodeDeploy 代理程式安裝的檔案 CodeDeploy 用戶端組態參考、和執行個體類型。
- 您要部署的執行個體可能沒有附加 IAM 執行個體設定檔，或者可能附加的 IAM 執行個體設定檔沒有必要的許可。

- 如果 IAM 執行個體設定檔未附加至您的執行個體，請建立具有所需權限的執行個體設定檔，然後將其連接。
- 如果 IAM 執行個體設定檔已附加至您的執行個體，請確定其具有必要的權限。

在您確定連接的執行個體描述檔已設定所需許可後，請重新啟動您的執行個體。如需詳細資訊，請參閱 步驟 4：為您的 Amazon EC2 執行個體建立 IAM 執行個體設定檔 Amazon EC2 使用者指南中的適用於 Amazon EC2 的 IAM 角色。

視窗 PowerShell 指令碼預設無法使用 64 位元版本 PowerShell 的視窗

如果做為部署一部分執行的 Windows PowerShell 指令碼依賴 64 位元功能 (例如，因為它消耗的記憶體超過 32 位元應用程式所允許的記憶體，或呼叫僅在 64 位元版本中提供的程式庫)，指令碼可能會損毀或無法如預期般執行。這是因為根據預設， CodeDeploy 會使用 32 位元版本的 Windows PowerShell 來執行屬於應用程式修訂版本一部分的 Windows PowerShell 指令碼。

在必須使用 64 位元版本的 Windows 執行的任何指令碼的開頭，新增如下所示的程式碼 PowerShell：

```
# Are you running in 32-bit mode?
#   (\SysWOW64\ = 32-bit mode)

if ($PSHOME -like "*SysWOW64*")
{
  Write-Warning "Restarting this script under 64-bit Windows PowerShell."

  # Restart this script under 64-bit Windows PowerShell.
  #   (\SysNative\ redirects to \System32\ for 64-bit mode)

  & (Join-Path ($PSHOME -replace "SysWOW64", "SysNative") powershell.exe) -File `
    (Join-Path $PSScriptRoot $MyInvocation.MyCommand) @args

  # Exit 32-bit script.

  Exit $LastExitCode
}

# Was restart successful?
Write-Warning "Hello from $PSHOME"
Write-Warning "  (\SysWOW64\ = 32-bit mode, \System32\ = 64-bit mode)"
Write-Warning "Original arguments (if any): $args"

# Your 64-bit script code follows here...
```

```
# ...
```

雖然此程式碼中的檔案路徑資訊看起來似乎違反直覺，但 32 位元 Windows 會 PowerShell 使用如下路徑：

```
c:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe
```

64 位元視窗 PowerShell 使用如下路徑：

```
c:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
```

Amazon ECS 部署問題疑難排解

主題

等待取代工作集時發生逾時

問題：使用部署 Amazon ECS 應用 CodeDeploy程式時，您看到下列錯誤訊息：

```
The deployment timed out while waiting for the replacement task set to
become healthy. This time out period is 60 minutes.
```

可能的原因：如果工作定義檔案或其他部署相關檔案中有錯誤，則可能會發生此錯誤。例如，如果您的任務定義檔案中的image欄位中有錯字，Amazon ECS 會嘗試擷取錯誤的容器映像並持續失敗，進而導致此錯誤。

可能的修復和後續步驟：

- 修正工作定義檔案和其他檔案中的印刷錯誤和組態問題。
- 查看相關的 Amazon ECS 服務事件，瞭解替換任務為何不會變得健康。如需 Amazon ECS 事件的詳細資訊，請參閱 Amazon 彈性容器服務開發人員指南中的 Amazon ECS 事件。
- 請查看 Amazon 彈性容器服務開發人員指南中的 Amazon ECS 疑難排解一節，瞭解與事件中訊息相關的錯誤。

等待通知繼續時發生逾時

問題：使用部署 Amazon ECS 應用 CodeDeploy程式時，您看到下列錯誤訊息：

```
The deployment timed out while waiting for a notification to continue. This time out period is n minutes.
```

可能的原因：如果您在建立部署群組時，在 [指定重新路由傳送流量的時間] 欄位中指定了等待時間，但是在等待時間到期之前無法完成部署，則可能會發生此錯誤。

可能的修復和後續步驟：

- 在您的部署群組中，設定指定何時將流量重新路由傳送至較大的時間並重新部署。如需詳細資訊，請參閱 為 Amazon ECS 部署 (主控台) 建立部署群組。
- 在您的部署群組中，變更指定何時重新路由傳送流量以立即重新路由傳送流量並重新部署。如需詳細資訊，請參閱 為 Amazon ECS 部署 (主控台) 建立部署群組。
- 重新部署，然後在--deployment-wait-type選項設定為READY_WAIT的情況下執行aws deploy continue-deployment AWS CLI 命令。請務必在指定重新路由傳送流量的時間到期中指定的時間之前執行此命令。

IAM 角色沒有足夠的許可

問題：使用部署 Amazon ECS 應用 CodeDeploy程式時，您看到下列錯誤訊息：

```
The IAM role role-arn does not give you permission to perform operations in the following AWS service: AWSLambda.
```

可能的原因：如果您在AppSpec 檔案Hooks區段中指定了 Lambda 函數，但未 CodeDeploy 授予 Lambda 服務權限，則可能會發生此錯誤。

可能的修正：將lambda:InvokeFunction權限新增至 CodeDeploy 服務角色。若要新增此權限，請將下列其中一個 AWS受管理的原則新增至角色：**AWSCodeDeployRoleForECS**或**AWSCodeDeployRoleForECSLimited**。如需有關這些原則以及如何將原則新增至 CodeDeploy 服務角色的資訊，請參閱步驟 2：建立服務角色 CodeDeploy。

等待狀態回呼時部署逾時

問題：使用部署 Amazon ECS 應用 CodeDeploy程式時，您看到下列錯誤訊息：

```
The deployment timed out while waiting for a status callback. CodeDeploy
expects a status callback within one hour after a deployment hook is
invoked.
```

可能的原因：如果您在AppSpec 檔案的Hooks區段中指定 Lambda 函數，可能會發生此錯誤，但 Lambda 函數無法呼叫必要的 PutLifecycleEventHookExecutionStatus API 來傳回Succeeded或Failed狀態 CodeDeploy。

可能的修復和後續步驟：

- 將codedeploy:putlifecycleEventHookExecutionStatus權限新增至您在 AppSpec 檔案中指定的 Lambda 函數所使用的 Lambda 執行角色。此權限授與 Lambda 函數將狀態傳回Succeeded或Failed的能力 CodeDeploy。如需 Lambda 執行角色的詳細資訊，請參閱AWS Lambda 使用者指南中的 Lambda 執行角色。
- 檢查您的 Lambda 函數程式碼和執行日誌，以確保您的 Lambda 函數正在呼叫 CodeDeploy的 PutLifecycleEventHookExecutionStatus API，以通 CodeDeploy知生命週期驗證測試Succeeded或Failed. 如需 putlifecycleEventHookExecutionStatus API 的相關資訊，請參閱 AWS CodeDeploy API 參考資料PutLifecycleEventHookExecutionStatus中的。如需 Lambda 執行日誌的相關資訊，請參閱存取 AWS Lambda. CloudWatch

部署失敗，因為一或多個生命週期事件驗證功能失敗

問題：使用部署 Amazon ECS 應用 CodeDeploy程式時，您看到下列錯誤訊息：

```
The deployment failed because one or more of the lifecycle event validation
functions failed.
```

可能的原因：如果您在AppSpec 檔案的Hooks區段中指定了 Lambda 函數，但在呼叫 Lambda 函數 CodeDeploy 時傳回，可Failed能會發生此錯誤PutLifecycleEventHookExecutionStatus。此失敗表示生命週期驗證測試失敗。 CodeDeploy

可能的下一個步驟：檢查 Lambda 執行日誌，瞭解驗證測試程式碼失敗的原因。如需 Lambda 執行日誌的相關資訊，請參閱存取 AWS Lambda. CloudWatch

ELB 因為下列錯誤而無法更新：主要工作集目標群組必須位於監聽器之後

問題：使用部署 Amazon ECS 應用 CodeDeploy程式時，您看到下列錯誤訊息：

```
The ELB could not be updated due to the following error: Primary taskset
target group must be behind listener
```

可能的原因：如果您已設定選擇性的測試監聽器，而且設定了錯誤的目標群組，就可能會發生此錯誤。如需有關中測試接聽程式的詳細資訊 CodeDeploy，請參閱在您開始 Amazon ECS 部署之前和Amazon ECS 部署期間會發生什麼情況。如需有關任務集的詳細資訊，請參閱 TaskSetAmazon 彈性容器服務 API 參考和describe-task-set AWS CLI命令參考的 Amazon ECS 一節。

可能的修正：確定彈性負載平衡的實際執行接聽程式和測試接聽程式都指向目前提供工作負載的目標群組。有三個地方可以檢查：

- 在 Amazon EC2 中，在負載平衡器的接聽程式和規則設定中。如需詳細資訊，請參閱《應用程式負載平衡器使用者指南》中的應用程式負載平衡器的接聽程式，或網路負載平衡器使用者指南中的網路負載平衡器的接聽程式。
- 在 Amazon ECS 中，在您的叢集中，在服務的聯網組態下。如需詳細資訊，請參閱 Amazon 彈性容器服務開發人員指南中的 Application Load Balancer 載平衡器和 Network Load Balancer 考量。
- 在中 CodeDeploy，在您的部署群組設定中。如需詳細資訊，請參閱 為 Amazon ECS 部署 (主控台) 建立部署群組。

使用 Auto Scaling 時，我的部署有時會失敗

問題：您正在搭配使用 Auto Scaling， CodeDeploy 且您注意到部署偶爾會失敗。如需有關此問題徵狀的詳細資訊，請參閱 Amazon Elastic Container Service 開發人員指南中設定為使用服務 auto auto 擴展的服務和藍/綠部署類型的主題，部署在部署期間不會遭到封鎖，但部署在某些情況下可能會失敗。

可能的原因：如果 CodeDeploy 和 Auto Scaling 程序發生衝突，可能會發生此問題

可能的修正：使用 RegisterScalableTarget API (或對應的register-scalable-target AWS CLI 命令) 在 CodeDeploy 部署期間暫停和繼續 Auto Scaling 程序。如需詳細資訊，請參閱《應用程式自動調整比例使用指南》中的「Application Auto Scaling 放」暫停和繼續

> **Note**
>
> CodeDeploy 不能RegisterScaleableTarget直接打電話。若要使用此 API，您必須設定 CodeDeploy 為傳送通知或事件至 Amazon 簡單通知服務 (或 Amazon CloudWatch)。然後，您必須設定 Amazon SNS (或 CloudWatch) 來呼叫 Lambda 函數，並將 Lambda 函數設定為呼叫 RegisterScalableTarget API。必須在將SuspendedState參數設定為的情況下呼叫 RegisterScalableTarget API，才true能暫停 Auto Scaling 作業並false繼續執行。當部署開始 (觸發 Auto Scaling 暫停作業)，或部署成功、失敗或停止 (以觸發 Auto Scaling 恢復作業) 時，必須發出通知或事件。 CodeDeploy
> 如需如何設定 CodeDeploy 以產生 Amazon SNS 通知或 CloudWatch事件的相關資訊，請參閱使用 Amazon CloudWatch 事件監控部署Monitoring Deployments with Amazon SNS Event Notifications。

只有 ALB 支援漸進式流量路由，建立/更新部署群組時，請改用 AllAtOnce 流量路由

問題：在中建立或更新部署群組時，您看到下列錯誤訊息 CodeDeploy：

```
Only ALB supports gradual traffic routing, use AllAtOnce Traffic routing
instead when you create/update Deployment group.
```

可能的原因：如果您使用 Network Load Balancer，並嘗試使用預先定義的部署設定，則可能會發生此錯誤CodeDeployDefault.ECSAllAtOnce。

可能的修正：

- 將預先定義的部署組態變更為CodeDeployDefault.ECSAllAtOnce。這是網路負載平衡器支援的唯一預先定義部署組態。

  如需預先定義的部署規劃的詳細資訊，請參閱適用於 Amazon ECS 運算平台的預先定義部署組態。
- 將負載平衡器變更為應用程式負載平衡器。應用程式負載平衡器支援所有預先定義的部署組態。如需建立 Application Load Balancer 的詳細資訊，請參閱為 CodeDeploy Amazon ECS 部署設定負載平衡器、目標群組和接聽程式。

即使我的部署成功，替換任務集也會失敗 Elastic Load Balancing 健康檢查，並且我的應用程序關閉

問題：即使 CodeDeploy 表示我的部署成功，替換任務集通過 Elastic Load Balancing 的健康狀態檢查失敗，並且我的應用程序已關閉。

可能的原因：如果您執行 CodeDeploy all-at-once 部署，而您的取代 (綠色) 工作集包含導致 Elastic Load Balancing 健康狀態檢查失敗的錯誤程式碼，可能會發生此問題。透過部 all-at-once 署組態，負載平衡器的健全狀況檢查會在流量轉移至該工作集之後 (也 CodeDeploy就是發生生`AllowTraffic`命週期事件之後)，開始在取代工作集上執行。這就是為什麼在交通轉移後，您會看到更換任務集的健康檢查失敗，但之前卻沒有。若要取得有關產生的 CodeDeploy 生命週期事件的資訊，請參閱Amazon ECS 部署期間會發生什麼情況。

可能的修正：

- 將部署組態從變更 all-at-once 為初期測試或線性。在初期測試或線性組態中，負載平衡器的健康狀態檢查會在取代環境中 CodeDeploy 安裝應用程式時，以及在流量轉移之前 (也就是在`Install`生命週期事件期間和事件發生之前) 開始在取代任務集上執行。`AllowTraffic`藉由允許檢查在應用程式安裝期間執行，但在流量轉移之前，將偵測到錯誤的應用程式程式碼，並在應用程式變成公開可用之前造成部署失敗。

  若要取得有關如何設定初期測試或線性部署的資訊，請參閱變更部署群組設定 CodeDeploy。

  如需 Amazon ECS 部署期間執行的 CodeDeploy 生命週期事件的相關資訊，請參閱Amazon ECS 部署期間會發生什麼情況。

  > **Note**
  >
  > Canary 和線性部署組態僅支援應用程式負載平衡器。

- 如果您想要保留 all-at-once 部署組態，請設定測試接聽程式，並使用`BeforeAllowTraffic`生命週期勾點檢查取代工作集的健全狀況狀態。如需詳細資訊，請參閱 Amazon ECS 部署的生命週期事件掛鉤清單。

我可以將多個負載平衡器附加到部署群組嗎？

沒有 如果您想要使用多個應用程式負載平衡器或網路負載平衡器，請使用 Amazon ECS 滾動式更新，而不是藍/綠部 CodeDeploy 署。如需有關滾動更新的詳細資訊，請參閱 Amazon 彈性容器服務開發人

員指南中的滾動更新。如需有關透過 Amazon ECS 使用多個負載平衡器的詳細資訊，請參閱 Amazon 彈性容器服務開發人員指南中的使用服務註冊多個目標群組。

如果沒有負載平衡器，是否可以執行 CodeDeploy 藍/綠部署？

否，您無法在沒有負載平衡器的情況下執行 CodeDeploy 藍/綠部署。如果您無法使用負載平衡器，請改用 Amazon ECS 的滾動式更新功能。如需 Amazon ECS 滾動式更新的詳細資訊，請參閱 Amazon 彈性容器服務開發人員指南中的滾動更新。

如何在部署期間使用新資訊更新我的 Amazon ECS 服務？

若要在執行部署時使用新參數 CodeDeploy 更新 Amazon ECS 服務，請在檔案的`resources`區段中指定參數。 AppSpec 僅支援少數 Amazon ECS 參數 CodeDeploy，例如任務定義檔案和容器名稱參數。如需可更新之 Amazon ECS 參數的完整清單 CodeDeploy ，請參閱 AppSpec Amazon ECS 部署的「資源」部分。

> **Note**
>
> 如果您需要使用不受支援的參數更新 Amazon ECS 服務 CodeDeploy，請完成以下任務：
>
> 1. 使用您想要更新的參數呼叫 Amazon ECS 的 `UpdateService` API。如需可更新的完整參數清單，請參閱 Amazon 彈性容器服務 API 參考UpdateService中的。
> 2. 若要將變更套用至任務，請建立新的 Amazon ECS 藍/綠部署。如需更多詳細資訊，請參閱建立 Amazon ECS 運算平台部署 (主控台)。

疑難排 AWS Lambda 部署問題

主題

- AWS Lambda 手動停止尚未設定復原的 Lambda 部署後，部署失敗

AWS Lambda 手動停止尚未設定復原的 Lambda 部署後，部署失敗

在某些情況下，部署中指定的 Lambda 函數的別名可能會參考兩個不同版本的函數。結果是後續嘗試部署 Lambda 函數失敗。如果 Lambda 部署未設定復原且手動停止，則 Lambda 部署可能會進入此狀態。若要繼續，請使用 AWS Lambda 主控台確定功能未設定為在兩個版本之間轉移流量：

1. 請登入 AWS Management Console 並開啟 AWS Lambda 主控台，網址為 https://console.aws.amazon.com/lambda/。
2. 從左側窗格中，選擇 Functions (函數)。
3. 選取 CodeDeploy 部署中的 Lambda 函數名稱。
4. 從別名中，選擇部 CodeDeploy 署中使用的別名，然後選擇編輯。
5. 從加權別名中，選擇**none**。這可確保不會將別名設定為將流量的百分比或權重轉移至多個版本。請記錄在 Version (版本) 中選取的版本。
6. 選擇儲存。
7. 開啟主 CodeDeploy 控台並嘗試部署步驟 5 中顯示在下拉式功能表中的版本。

對部署群組問題進行故障診斷

將執行個體加入標籤做為部署群組的一部分，不會自動將應用程式部署至新執行個體

CodeDeploy 不會自動將您的應用程式部署到新標記的執行個體。您必須在部署群組中建立新的部署。

您可以用 CodeDeploy 來啟用自動部署到 Amazon EC2 自 Auto Scaling 群組中的新 EC2 執行個體。如需詳細資訊，請參閱 CodeDeploy 與 Amazon EC2 Auto Scaling 集成。

對執行個體問題進行故障診斷

主題

- 標籤必須正確設定
- AWS CodeDeploy 代理程式必須安裝並在執行個體上執行
- 如果執行個體在部署期間終止，在最多一小時內部署不會失敗
- 分析日誌檔案以調查執行個體的部署失敗
- 如果意外刪除了新的 CodeDeploy 記錄檔，請建立新的記錄檔
- 疑難排解「InvalidSignatureException — 簽章已過期:[時間] 現在早於 [時間]」部署錯誤

標籤必須正確設定

使用指list-deployment-instances令確認用於部署的執行個體已正確標記。如果輸出中缺少 EC2 執行個體，請使用 EC2 主控台確認已在執行個體上設定標籤。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的主控台中的使用標籤。

> **Note**
>
> 如果您標記執行個體並立即用 CodeDeploy 來部署應用程式，則該執行個體可能不會包含在部署中。這是因為可能需要幾分鐘才 CodeDeploy 能讀取標籤。建議您在為執行個體套用標籤與嘗試對執行個體進行部署之間等待至少五分鐘。

AWS CodeDeploy 代理程式必須安裝並在執行個體上執行

若要確認 CodeDeploy 代理程式是否已安裝並在執行個體上執行，請參閱確認 CodeDeploy 代理程式正在執行。

若要安裝、解除安裝或重新安裝 CodeDeploy 代理程式，請參閱安裝 CodeDeploy 代理程式。

如果執行個體在部署期間終止，在最多一小時內部署不會失敗

CodeDeploy 為每個執行到完成的部署生命週期事件提供一個小時的時間。這為長時間執行的指令碼提供足夠的時間。

如果生命週期事件正在進行時指令碼未執行完成 (例如，如果執行個體終止或 CodeDeploy 代理程式關閉)，則部署狀態最多可能需要一個小時才會顯示為「失敗」。即使在指令碼中指定的逾時期間短於一小時，也會發生此情況。這是因為執行個體終止時， CodeDeploy 代理程式會關閉，而且無法處理更多指令碼。

如果執行個體在生命週期事件之間或在第一個生命週期事件步驟開始之前終止，逾時將在五分鐘後發生。

分析日誌檔案以調查執行個體的部署失敗

如果部署中的執行個體具有 Succeeded 以外的任何狀態，您可以檢閱部署日誌檔案資料來協助識別問題。如需存取部署日誌資料的詳細資訊，請參閱檢視 CodeDeploy EC2/內部部署的記錄資料。

如果意外刪除了新的 CodeDeploy 記錄檔，請建立新的記錄檔

如果您不小心刪除執行個體上的部署記錄檔，則 CodeDeploy 不會建立取代記錄檔。若要建立新的日誌檔案，請登入執行個體，然後執行這些命令：

對於 Amazon Linux、Ubuntu 伺服器或 RHEL 執行個體，請按以下順序執行以下命令，一次執行一個命令：

```
systemctl stop codedeploy-agent
```

```
systemctl start codedeploy-agent
```

對於視窗伺服器執行個體：

```
powershell.exe -Command Restart-Service -Name codedeployagent
```

疑難排解「InvalidSignatureException — 簽章已過期:[時間] 現在早於 [時間]」部署錯誤

CodeDeploy 需要準確的時間參考才能執行其操作。如果執行個體上的日期和時間設定不正確，它們可能與 CodeDeploy 拒絕部署要求的簽章日期不符。

若要避免與時間設定不正確相關的部署失敗，請參閱下列主題：

- 設定您 Linux 執行個體的時間
- 設定 Windows 執行個體的時間

問題疑 GitHub 難排解

無效的 GitHub OAuth 令牌

CodeDeploy 2017 年 6 月後建立的應用程式會針對每個 AWS 區域使用 GitHub OAuth 權杖。使用綁定到特定 AWS 區域的令牌使您可以更好地控制哪些 CodeDeploy 應用程序可以訪問 GitHub 存儲庫。

如果您收到 GitHub 令牌錯誤，則可能有一個現在無效的舊令牌。

若要修正無效的 GitHub OAuth 權杖

1. 使用下列其中一種方法移除舊權杖：
   - 若要使用 API 移除舊權杖，請使用 DeleteGitHubAccountToken.
   - 要使用以下命令刪除舊令牌 AWS Command Line Interface：
     a. 前往權杖所在的電腦。
     b. 請確定 AWS CLI 已安裝在此電腦上。如需安裝指示，請參閱《AWS Command Line Interface 使用指南》 AWS CLI中的〈安裝、更新和解除安裝〉
     c. 在權杖所在的電腦上輸入下列指令：

        ```
        aws delete-git-hub-account-token
        ```

        如需有關命令語法的詳細資訊，請參閱 delete-git-hub-account-token。
2. 新增新的 OAuth 字符。如需詳細資訊，請參閱 CodeDeploy 與整合 GitHub。

超過 GitHub OAuth 令牌的最大數量

建立 CodeDeploy 部署時，允許的 GitHub 權杖數目上限為 10。如果您收到有關 GitHub OAuth 令牌的錯誤，請確保您有 10 個或更少的令牌。如果您有超過 10 個字符，則最初建立的字符將會無效。例如，如果您有 11 個字符，您建立的第一個字符將會無效。如果您有 12 個字符，您建立的前兩個字符將會無效。如需使用 CodeDeploy API 移除舊權杖的相關資訊，請參閱 DeleteGitHubAccountToken。

疑難排解 Amazon EC2 Auto Scaling 問題

主題

- 一般的 Amazon EC2 Auto Scaling 故障排除
- 「CodeDeployRole 不授予您在以下 AWS 服務中執行操作的權限： AmazonAutoScaling」錯誤
- Amazon EC2 Auto Scaling 群組中的執行個體會持續佈建和終止，然後才能部署修訂
- 終止或重新啟動 Amazon EC2 Auto Scaling 執行個體可能會導致部署失敗
- 避免將多個部署群組與單一 Amazon EC2 Auto Scaling 群組建立關聯
- Amazon EC2 Auto Scaling 群組中的 EC2 執行個體無法啟動並收到錯誤「活動訊號逾時」
- 不匹配的 Amazon EC2 自 Auto Scaling 生命週期掛鉤可能會導致對 Amazon EC2 自動 Auto Scaling 群組的自動部署停止或失敗
- 「部署失敗，因為找不到您的部署群組的執行個體」錯誤

一般的 Amazon EC2 Auto Scaling 故障排除

部署至 Amazon EC2 Auto Scaling 群組中 EC2 執行個體的部署可能會失敗，原因如下：

- Amazon EC2 Auto Scaling 會持續啟動和終止 EC2 執行個體。如果 CodeDeploy 無法自動部署應用程式修訂版，Amazon EC2 Auto Scaling 會持續啟動和終止 EC2 執行個體。

  取消 Amazon EC2 Auto Scaling 群組與 CodeDeploy 部署群組的關聯，或變更 Amazon EC2 Auto Scaling 群組的組態，使所需的執行個體數目符合目前執行個體的數量 (因此防止 Amazon EC2 Auto Scaling 啟動任何更多 EC2 執行個體)。如需詳細資訊，請參閱 Amazon EC2 自動擴展的變更部署群組設定 CodeDeploy或手動擴展。
- CodeDeploy 代理程式沒有回應。如果在 EC2 執行個體啟動或啟動後立即執行的初始化指令碼 (例如 cloud init 指令碼) 需要一個多小時才能執行，則可能不會安裝 CodeDeploy 代理程式。 CodeDeploy 有一個小時的逾時， CodeDeploy 代理程式回應擱置的部署。若要解決此問題，請將您的初始化指令碼移至 CodeDeploy應用程式修訂版本。
- Amazon EC2 Auto Scaling 群組中的 EC2 執行個體會在部署期間重新啟動。如果 EC2 執行個體在部署期間重新開機，或在處理部署命令時關閉 CodeDeploy 代理程式，則部署可能會失敗。如需詳細資訊，請參閱 終止或重新啟動 Amazon EC2 Auto Scaling 執行個體可能會導致部署失敗。
- 多個應用程式修訂會同時部署到 Amazon EC2 Auto Scaling 群組中的相同 EC2 執行個體。如果其中一個部署的指令碼執行時間超過幾分鐘，則同時將多個應用程式修訂部署到 Amazon EC2 Auto Scaling 群組中的相同 EC2 執行個體可能會失敗。請勿將多個應用程式修訂部署到 Amazon EC2 Auto Scaling 群組中的相同 EC2 執行個體。
- 作為 Amazon EC2 自動擴展群組一部分啟動的新 EC2 執行個體，部署失敗。在這個案例中，在部署中執行指令碼可能會阻止在 Amazon EC2 自動擴展群組中啟動 EC2 執行個體。(Amazon EC2 Auto Scaling 群組中的其他 EC2 執行個體可能看似正常執行。) 若要解決這個問題，請確保先完成所有其他指令碼：
  - CodeDeploy AMI 中未包含代理程式：如果您在啟動新執行個體時使用cfn-init指令來安裝 CodeDeploy 代理程式，請將代理程式安裝指令碼放在 AWS CloudFormation 範本`cfn-init`區段的末尾。
  - CodeDeploy 代理程式包含在 AMI 中：設定 AMI，讓代理程式在建立執行個體時處於`Stopped`狀態，然後在指令碼程式庫中包含用於啟動代理程式的`cfn-init`指令碼。

「CodeDeployRole 不授予您在以下 AWS 服務中執行操作的權限：AmazonAutoScaling」錯誤

使用以啟動範本建立的 Auto Scaling 群組的部署需要下列權限。這些是受AWSCodeDeployRole AWS 管理策略所授與的權限之外的補充。

- EC2:RunInstances
- EC2:CreateTags
- iam:PassRole

如何缺少這些許可，您可能會收到此錯誤。如需詳細資訊教學課程：用 CodeDeploy 於將應用程式部署到 Auto Scaling 群組，請參閱為 Auto Scaling 群組建立啟動範本和 Amazon EC2 Au to Scaling 使用者指南中的許可。

Amazon EC2 Auto Scaling 群組中的執行個體會持續佈建和終止，然後才能部署修訂

在某些情況下，錯誤可能會導致無法成功部署到 Amazon EC2 Auto Scaling 群組中新佈建的執行個體。因此沒有執行正常的執行個體，也沒有成功的部署。由於部署無法成功執行或成功完成，因此執行個體在建立後很快就會終止。然後，Amazon EC2 Auto Scaling 群組組態會佈建另一批執行個體嘗試滿足運作狀態最低的主機需求。這個批次也會終止，此循環會繼續下去。

可能的原因包括：

- Amazon EC2 Auto Scaling 群組運作狀態檢查失敗。
- 應用程式修訂中的錯誤。

若要解決此問題，請遵循這些步驟：

1. 手動建立不屬於 Amazon EC2 Auto Scaling 群組的 EC2 執行個體。使用唯一的 EC2 執行個體標籤，為執行個體加入標籤。
2. 將新執行個體新增至受影響的部署群組。
3. 將沒有錯誤的新應用程式修訂部署至部署群組。

這會提示 Amazon EC2 Auto Scaling 群組將應用程式修訂部署到 Amazon EC2 Auto Scaling 群組中的 future 執行個體。

> **Note**
>
> 確認部署成功後，請刪除您建立的執行個體，以避免 AWS 帳戶的持續費用。

終止或重新啟動 Amazon EC2 Auto Scaling 執行個體可能會導致部署失敗

如果 EC2 執行個體透過 Amazon EC2 Auto Scaling 啟動，然後該執行個體終止或重新啟動，則對該執行個體的部署可能會因下列原因而失敗：

- 在進行中的部署期間，擴展事件或任何其他終止事件會導致執行個體從 Amazon EC2 Auto Scaling 群組中分離，然後終止。由於部署無法完成，因此將會失敗。
- 執行個體會重新啟動，但啟動執行個體需要五分鐘以上的時間。 CodeDeploy 將此視為逾時。該服務會使所有目前和未來對該執行個體的部署失敗。

解決此問題：

- 一般而言，請確保在終止或重新啟動執行個體之前完成所有部署。請確保在執行個體啟動或重新啟動後，啟動所有部署。
- 如果您為 Amazon Amazon EC2 Auto Scaling 組態指定 Windows 伺服器基礎的亞馬遜機器映像 (AMI)，並使用 EC2Config 服務來設定執行個體的電腦名稱，則部署可能會失敗。若要修正此問題，請在 Windows 伺服器基礎 AMI 的 EC2 服務屬性的 [一般] 索引標籤上，清除 [設定電腦名稱]。清除此核取方塊後，使用該 Windows 伺服器基礎 AMI 啟動的所有新 Windows 伺服器 Amazon EC2 Auto Scaling 執行個體都會停用此行為。對於啟用此行為的 Windows 伺服器 Amazon EC2 Auto Scaling 執行個體，您不需要清除此核取方塊。只需在重新啟動後，將失敗的部署重新部署至這些執行個體。

避免將多個部署群組與單一 Amazon EC2 Auto Scaling 群組建立關聯

最佳實務是，您應該只將一個部署群組與每個 Amazon EC2 Auto Scaling 群組建立關聯。

這是因為如果 Amazon EC2 Auto Scaling 擴展具有與多個部署群組相關聯掛接的執行個體，它會一次傳送所有勾點的通知。這會導致每個執行個體的多個部署同時啟動。當多個部署同時傳送命令至 CodeDeploy 代理程式時，可能會達到生命週期事件與部署開始或前一個生命週期事件結束之間的五分鐘逾時。如果發生這種情況，部署即會失敗，即使部署程序如預期執行。

> **Note**
>
> 生命週期事件中指令碼的預設逾時時間為 30 分鐘。您可以將逾時變更為 AppSpec 檔案中的其他值。如需詳細資訊，請參閱 新增EC2/ AppSpec 內部部署的檔案。

如果同時出現一個以上的部署嘗試，當部署發生時，會無法控制順序。

最後，如果部署到任何執行個體失敗，Amazon EC2 Auto Scaling 會立即終止執行個體。當第一個執行個體關閉時，其他執行中的部署開始失敗。由於 CodeDeploy 代理程式會 CodeDeploy 有一小時的逾時時間回應擱置中的部署，因此每個執行個體最多可能需要 60 分鐘的時間逾時。

如需 Amazon EC2 Auto Scaling 的詳細資訊，請參閱引擎蓋之下： CodeDeploy 和 Auto Scaling 整合。

Amazon EC2 Auto Scaling 群組中的 EC2 執行個體無法啟動並收到錯誤「活動訊號逾時」

Amazon EC2 Auto Scaling 群組可能無法啟動新的 EC2 執行個體，產生類似下列內容的訊息：

```
Launching a new EC2 instance <instance-Id>. Status Reason: Instance failed
to complete user's Lifecycle Action: Lifecycle Action with token<token-Id>
was abandoned: Heartbeat Timeout.
```

此訊息通常會指出下列項目之一：

- 已達到與 AWS 帳戶相關聯的同時部署數目上限。如需部署限制的詳細資訊，請參閱 CodeDeploy 配額。
- Auto Scaling 組嘗試過快地啟動太多 EC2 執行個體。每個新執行個體的 API 呼叫RecordLifecycleActionHeartbeat或CompleteLifecycleAction針對每個新執行個體進行了限制。
- 中的應用程式 CodeDeploy 在更新或刪除其關聯的部署群組之前已刪除。

  刪除應用程式或部署群組時，會 CodeDeploy 嘗試清除與該應用程式或部署群組相關聯的任何 Amazon EC2 Auto Scaling 掛鉤，但可能會保留一些掛鉤。如果您執行命令來刪除部署群組，輸出中會傳回剩餘關聯。不過，如果您執行命令來刪除應用程式，剩餘關聯不會在輸出中出現。

  因此，做為最佳實務，您應該先刪除所有與應用程式建立關聯的部署群組，再刪除應用程式。您可以使用命令輸出，來識別必須手動刪除的生命週期關聯。

如果您收到「Heartbeat Timeout」錯誤訊息，則可以透過執行下列操作來判斷剩餘的生命週期關聯是否為原因，並解決問題：

1. 執行以下任意一項：

   - 呼叫命delete-deployment-group令以刪除與導致活動訊號逾時之 Auto Scaling 群組相關聯的部署群組。
   - 使用非空白的 Auto Scaling 群組名稱清單呼叫update-deployment-group指令，以卸離所有CodeDeploy受管理的 Auto Scaling 生命週期掛接。

     例如，輸入下列 AWS CLI 命令：

     ```
     aws deploy update-deployment-group --application-name my-example-app
     --current-deployment-group-name my-deployment-group --auto-scaling-
     groups
     ```

     作為另一個例子，如果您將 CodeDeploy API 與 Java 一起使用，請調用UpdateDeploymentGroup並設置autoScalingGroups為new ArrayList<String>()。這autoScalingGroups將設置為一個空列表並刪除現有列表。不要使用null，這是默認的，因為這會保留autoScalingGroups原樣，這不是你想要的。

   檢查呼叫的輸出。如果輸出包含具有 Amazon EC2 Auto Scaling 生命週期勾點清單的hooksNotCleanedUp結構，則有剩餘的生命週期掛鉤。

2. 呼叫命describe-lifecycle-hooks令，指定與無法啟動的 EC2 執行個體相關聯的 Amazon EC2 自動擴展群組名稱。在輸出中，尋找下列任一項目：

   - Amazon EC2 Auto Scaling 生命週期勾點名稱與您在步驟 1 中識別的hooksNotCleanedUp結構相對應。
   - Amazon EC2 Auto Scaling 生命週期勾點名稱，其中包含與失敗的 Auto Scaling 群組相關聯的部署群組名稱。
   - Amazon EC2 Auto Scaling 生命週期勾點名稱可能導致部 CodeDeploy 署的活動訊號逾時。

3. 如果勾點屬於步驟 2 中列出的其中一個類別，請呼叫delete-lifecycle-hook指令將其刪除。在呼叫中指定 Amazon EC2 Auto Scaling 群組和生命週期勾點。

> **Important**
>
> 只刪除導致問題的掛接，如步驟 2 所述。如果刪除可行的掛鉤，則部署可能會失敗，或者可 CodeDeploy 能無法部署應用程式修訂版以向外擴充 EC2 執行個體。

4. 使用所需的「Auto Scaling」群組名稱呼叫update-deployment-group或create-deployment-group指令。 CodeDeploy使用新的 UUID 重新安裝「Auto Scaling」掛接。

> **Note**
>
> 如果您從 CodeDeploy 部署群組中分離 Auto Scaling 群組，則任何進行中的 Auto Scaling 群組部署可能會失敗，而由 Auto Scaling 群組向外擴充的新 EC2 執行個體將不會從中 CodeDeploy 收到應用程式修訂版本。若要讓 Auto Scaling 再次運作 CodeDeploy，您需要將 Auto Scaling 群組重新連接至部署群組，然後呼叫新的`CreateDeployment`來啟動叢集範圍的部署。

不匹配的 Amazon EC2 自 Auto Scaling 生命週期掛鉤可能會導致對 Amazon EC2 自動 Auto Scaling 群組的自動部署停止或失敗

Amazon EC2 Auto Scaling 並 CodeDeploy 使用生命週期勾點來判斷哪些應用程式修訂版在 Amazon EC2 自動擴展群組中啟動後，應將哪些應用程式修訂部署到哪些 EC2 執行個體。如果生命週期掛鉤和有關這些掛鉤的資訊在 Amazon EC2 Auto Scaling 和中不完全相符，則自動部署可能會停止或失敗 CodeDeploy。

如果部署到 Amazon EC2 Auto Scaling 群組失敗，請查看 Amazon EC2 自 Auto Scaling 中的生命週期勾點名稱是否 CodeDeploy 相符。如果沒有，請使用這些 AWS CLI 命令呼叫。

首先，取得 Amazon EC2 Auto Scaling 群組和部署群組的生命週期勾點名稱清單：

1. 呼叫命describe-lifecycle-hooks令，指定與中部署群組相關聯的 Amazon EC2 Auto Scaling 群組的名稱 CodeDeploy。在輸出的 `LifecycleHooks` 清單中，記錄每個 `LifecycleHookName` 值。
2. 呼叫命get-deployment-group令，指定與 Amazon EC2 Auto Scaling 群組關聯的部署群組名稱。在輸出的`autoScalingGroups`清單中，找出名稱值與 Amazon EC2 Auto Scaling 群組名稱相符的每個項目，然後記下對應的hook值。

現在比較兩組生命週期關聯的名稱。如果完全相符 (字元對字元)，那麼這就不是問題所在。您可能想要嘗試本節其他部分描述的其他 Amazon EC2 Auto Scaling 疑難排解步驟。

不過，如果兩組生命週期關聯名稱不完全相符 (字元對字元)，請執行下列作業：

1. 如果在 describe-lifecycle-hooks 命令輸出中包含在 get-deployment-group 命令輸出中未包含的生命週期關聯名稱，則執行以下作業：

   a. 針對describe-lifecycle-hooks命令輸出中的每個生命週期掛接名稱，呼叫delete-lifecycle-hook命令。

   b. 呼叫命update-deployment-group令，指定原始 Amazon EC2 Auto Scaling 群組的名稱。CodeDeploy 在 Amazon EC2 Auto Scaling 群組中建立新的替代生命週期掛鉤，並將生命週期勾點與部署群組建立關聯。現在應該會在新執行個體新增至 Amazon EC2 自動 Auto Scaling 群組時恢復自動部署。

2. 如果在 get-deployment-group 命令輸出中包含在 describe-lifecycle-hooks 命令輸出中未包含的生命週期關聯名稱，則執行以下作業：

   a. 呼叫命update-deployment-group令，但不要指定原始 Amazon EC2 Auto Scaling 群組的名稱。

   b. 再次呼叫update-deployment-group命令，但這次要指定原始 Amazon EC2 Auto Scaling 群組的名稱。 CodeDeploy 在 Amazon EC2 Auto Scaling 群組中重新建立遺失的生命週期掛鉤。現在應該會在新執行個體新增至 Amazon EC2 自動 Auto Scaling 群組時恢復自動部署。

在您取得兩組生命週期勾點名稱完全符合之後，字元的字元應該再次部署應用程式修訂，但只能部署到新執行個體，因為這些執行個體新增至 Amazon EC2 Auto Scaling 群組。已經在 Amazon EC2 自動擴展群組中的執行個體不會自動進行部署。

「部署失敗，因為找不到您的部署群組的執行個體」錯誤

如果您看到下列 CodeDeploy 錯誤，請閱讀本節：

```
The deployment failed because no instances were found for your deployment
group. Check your deployment group settings to make sure the tags for your
EC2 instances or Auto Scaling groups correctly identify the instances you
want to deploy to, and then try again.
```

此錯誤的可能原因包括：

1. 您的部署群組設定包含 EC2 執行個體、現場部署執行個體或 Auto Scaling 群組不正確的標記。若要修正此問題，請檢查您的標籤是否正確，然後重新部署應用程式。
2. 您的叢集會在部署開始後向外擴充。在這個案例中，您會看到叢集中InService狀態良好的執行個體，但您也會看到上述錯誤。若要修正此問題，請重新部署您的應用程式。
3. 您的「Auto Scaling」群組不包含任何處於該InService狀態的執行個體。在這個案例中，當您嘗試執行叢集範圍部署時，部署會失敗並顯示上述錯誤訊息，因為至少 CodeDeploy 需要一個執行個體處於狀態InService。InService狀態中可能沒有執行個體的原因有很多。其中一些包括：
   - 您已排程 (或手動設定)「Auto Scaling」群組大小為0。
   - Auto Scaling 檢測到錯誤的 EC2 實例（例如，EC2 實例出現硬件故障），因此將它們全部取消，不留在InService狀態中。
   - 在從0到向外延展事件期間1，部 CodeDeploy 署了先前成功的修訂版 (稱為上次成功的修訂版)，該修訂版本自上次部署後變得不健康。這會造成向外延展執行個體上的部署失敗，進而導致 Auto Scaling 取消執行個體，而不會在狀態中留下執行個體。InService

     如果您發現InService狀態中沒有執行個體，請依照下列程序所述疑難排解問題To troubleshoot the error if there are no instances in the InService state。

若要疑難排解 InService 狀態中沒有執行個體時的錯誤

1. 在 Amazon EC2 主控台中，確認所需容量設定。如果為零，請將其設定為正數。等待執行個體為InService，這表示部署成功。您已修正此問題，並且可以略過此疑難排解程序的剩餘步驟。如需有關設定所需容量設定的資訊，請參閱 Amazon EC2 Auto Scaling 使用者指南中的設定 Auto Scaling 群組的容量限制。
2. 如果 Auto Scaling 持續嘗試啟動新的 EC2 執行個體以符合所需容量，但永遠無法滿足向外擴充，通常是因為 Auto Scaling 生命週期掛鉤失敗。疑難排解此問題，如下所示：

   a. 若要檢查哪個 Auto Scaling 生命週期勾點事件失敗，請參閱 Amazon EC2 Auto Scaling 使用者指南中的「驗證自動擴展」群組的擴展活動。

   b. 如果失敗掛接的名稱是CodeDeploy-managed-automatic-launch-deployment-hook-*DEPLOYMENT_GROUP_NAME*，請移至 CodeDeploy，尋找部署群組，然後尋找 Auto Scaling 啟動的失敗部署。然後調查部署失敗的原因。

   c. 如果您瞭解部署失敗的原因 (例如，發生 CloudWatch 警示)，並且您可以在不變更修訂版本的情況下修正問題，請立即進行。

   d. 如果經過調查後，您判斷上次成功的 CodeDeploy修訂已不再健康狀態，且 Auto Scaling 群組中沒有健康狀態良好的執行個體，則表示您處於部署死結案例中。若要解決此問題，您必須暫

時從 Auto Scaling 群組中移除 CodeDeploy生命週期勾點，然後重新安裝勾點並重新部署新的 (良好) 修訂，藉此修正錯誤 CodeDeploy 的修訂。如需說明，請參閱：

- To fix the deployment deadlock issue (CLI)
- To fix the deployment deadlock issue (console)

若要修正部署鎖死問題 (CLI)

1. (選擇性) 封鎖造成 CodeDeploy 錯誤的 CI/CD 管線，以便在修正此問題時不會發生非預期的部署。
2. 記下您目前的「Auto Scaling」DesiredCapacity設定：

   ```
   aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name ASG_NAME
   ```

   您可能需要在此程序結束時縮放回此數字。
3. 將「Auto Scaling」DesiredCapacity設定設定為1。如果您想要的容量大於1開始，則這是可選的。藉由將其減少為1，執行個體稍後佈建和部署的時間將較少，進而加快疑難排解的速度。如果您的 Auto Scaling 所需容量原本設定為0，則必須將其增加為1。這是強制性的。

   AWS 自動擴展 set-desired-capacity --auto-scaling-group-name *ASG* 名稱-所需容量 1

   > **Note**
   >
   > 此程序的其餘步驟假設您已將設定DesiredCapacity為1。

   此時，「Auto Scaling」會嘗試縮放至一個執行個體。然後，由於新 CodeDeploy 增的勾點仍然存在，所以 CodeDeploy 嘗試部署；部署失敗；Auto Scaling 會取消執行個體；Auto Scaling 會嘗試重新啟動執行個體以達到所需容量的其中一個執行個體，而且會再次失敗。您正處於取消重新啟動迴圈中。
4. 從部署群組中取消註冊 Auto Scaling 群組：

> **Warning**
>
> 以下命令將啟動一個沒有軟件的新 EC2 實例。執行指令之前，請確定未執行任何軟體的 Auto Scaling `InService` 執行個體是可接受的。例如，確定與執行個體相關聯的負載平衡器不會在沒有軟體的情況下將流量傳送到此主機。

> **Important**
>
> 使用如下所示的 CodeDeploy 指令移除掛鉤。請勿透過 Auto Scaling 服務移除勾點，因為移除作業將無法被辨識 CodeDeploy。

```
aws deploy update-deployment-group --application-name APPLICATION_NAME
--current-deployment-group-name DEPLOYMENT_GROUP_NAME --auto-scaling-
groups
```

執行此命令後，會發生下列情況：

a. CodeDeploy 從部署群組取消註冊「Auto Scaling」群組。

b. CodeDeploy 從「Auto Scaling」群組中移除「Auto Scaling」生命週期勾

c. 由於導致部署失敗的掛鉤不再存在，Auto Scaling 會取消現有 EC2 執行個體，並立即啟動新的執行個體以擴展到所需的容量。新執行個體應該很快就會進入`InService`狀態。新執行個體不包含軟體。

5. 等待 EC2 實例進入狀`InService`態。若要驗證其狀態，請使用下列命令：

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names
ASG_NAME --query AutoScalingGroups[0].Instances[*].LifecycleState
```

6. 將鉤子添加回 EC2 實例：

> **Important**
>
> 使用如下所示的 CodeDeploy 指令加入掛鉤。請勿使用 Auto Scaling 服務來新增勾點，因為新增功能將無法被辨識 CodeDeploy。

```
aws deploy update-deployment-group --application-name APPLICATION_NAME
--current-deployment-group-name DEPLOYMENT_GROUP_NAME --auto-scaling-
groups ASG_NAME
```

執行此命令後，會發生下列情況：

a. CodeDeploy 將 Auto Scaling 生命週期掛鉤重新安裝到 EC2 執行個體

b. CodeDeploy 將「Auto Scaling」群組與部署群組重新註冊。

7. 使用 Amazon S3 建立叢集部署，或您知道健康狀況良好且想要使用的 GitHub 修訂版。

例如，如果修訂版是 Amazon S3 儲存貯體中使用物件金鑰呼叫`my-revision-bucket`的 .zip 檔案`httpd_app.zip`，請輸入下列命令：

```
aws deploy create-deployment --application-name APPLICATION_NAME
--deployment-group-name DEPLOYMENT_GROUP_NAME --
revision "revisionType=S3,s3Location={bucket=my-revision-
bucket,bundleType=zip,key=httpd_app.zip}"
```

由於 Auto Scaling 群組中現在有一個`InService`執行個體，因此此部署應該可以運作，而且您應該不會再看到錯誤部署失敗，因為找不到您的部署群組的執行個體。

8. 部署成功後，如果您先前將 Auto Scaling 群組擴展至原始容量：

```
aws autoscaling set-desired-capacity --auto-scaling-group-name ASG_NAME
--desired-capacity ORIGINAL_CAPACITY
```

若要修正部署鎖死問題 (主控台)

1. (選擇性) 封鎖造成 CodeDeploy 錯誤的 CI/CD 管線，以便在修正此問題時不會發生非預期的部署。

2. 前往 Amazon EC2 主控台，記下您的 Auto Scaling 所需容量設定。您可能需要在此程序結束時縮放回此數字。如需尋找此設定的相關資訊，請參閱在 Auto Scaling 群組上設定容量限制。

3. 將所需的 EC2 執行個體數量設定為`1`：

如果您想要的容量大於`1`開始，則這是可選的。藉由將其減少為`1`，執行個體稍後佈建和部署的時間將較少，進而加快疑難排解的速度。如果您的 Auto Scaling 所需容量原本設定為`0`，您必須將其增加為`1`。這是強制性的。

> **Note**
>
> 此程序的其餘步驟假設您已將 [想要的容量] 設定為1。

a. 前往網址 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。

b. 選擇適當的「區域」。

c. 轉到有問題的「Auto Scaling」組。

d. 在群組詳細資料中，選擇編輯。

e. 將 [所需容量] 設定為**1**。

f. 選擇更新。

4. 從部署群組中取消註冊 Auto Scaling 群組：

> **Warning**
>
> 以下子步驟將啟動不含軟體的新 EC2 執行個體。執行指令之前，請確定未執行任何軟體的 Auto Scaling `InService` 執行個體是可接受的。例如，確定與執行個體相關聯的負載平衡器不會在沒有軟體的情況下將流量傳送到此主機。

a. 開啟主 CodeDeploy 控台，網址為 https://console.aws.amazon.com/codedeploy/。

b. 選擇適當的「區域」。

c. 在導覽窗格中，選擇 Applications (應用程式)。

d. 選擇您的 CodeDeploy 應用程序的名稱。

e. 選擇部 CodeDeploy 署群組的名稱。

f. 選擇編輯。

g. 在環境組態中，取消選取 Amazon EC2 Auto Scaling 群組。

> **Note**
>
> 如果沒有定義環境配置，則控制台不允許您保存配置。若要略過檢查，請暫時新增`EC2`或您`On-premises`知道不會解析為任何主機的標籤。若要新增標籤，請選

取 Amazon EC2 執行個體或現場部署執行個體，然後新增**EC2**或的標籤金鑰**On-premises**。您可以將標籤值保留為空。

h. 選擇儲存變更。

完成這些子步驟後，會發生下列情況：

i. CodeDeploy 從部署群組取消註冊「Auto Scaling」群組。

ii. CodeDeploy 從「Auto Scaling」群組中移除「Auto Scaling」生命週期勾

iii. 由於導致部署失敗的掛鉤不再存在，Auto Scaling 會取消現有 EC2 執行個體，並立即啟動新的執行個體以擴展到所需的容量。新執行個體應該很快就會進入`InService`狀態。新執行個體不包含軟體。

5. 等待 EC2 實例進入狀`InService`態。若要驗證其狀態：

a. 前往 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。

b. 在導覽窗格中，選擇 Auto Scaling Groups (AS 安全群組)。

c. 選擇您的「Auto Scaling」群組。

d. 在內容窗格中，選擇「執行處理管理」索引標籤。

e. 在「執行個體」下，確定「生命週期」欄指示在執行個體InService旁邊。

6. 使用您用來移除該群組的相同方法，將 Auto Scaling 群組重新註冊至 CodeDeploy 部署群組：

a. 開啟主 CodeDeploy 控台，網址為 https://console.aws.amazon.com/codedeploy/。

b. 選擇適當的「區域」。

c. 在導覽窗格中，選擇 Applications (應用程式)。

d. 選擇您的 CodeDeploy 應用程序的名稱。

e. 選擇部 CodeDeploy 署群組的名稱。

f. 選擇編輯。

g. 在環境組態中，選取 Amazon EC2 Auto Scaling 群組，然後從清單中選取您的 Auto Scaling 群組。

h. 在 Amazon EC2 執行個體或現場部署執行個體下，找到您新增的標籤並將其移除。

i. 取消選取 Amazon EC2 執行個體或現場部署執行個體旁邊的核取方塊。

j. 選擇儲存變更。

此設定會將生命週期勾點重新安裝到 Auto Scaling 群組中。

7. 使用 Amazon S3 建立叢集部署，或您知道健康狀況良好且想要使用的 GitHub 修訂版。

   例如，如果修訂版是`my-revision-bucket`使用物件金鑰呼叫的 Amazon S3 儲存貯體中的 .zip 檔案`httpd_app.zip`，請執行下列動作：

   a. 在 CodeDeploy 主控台的 [部署群組] 頁面中，選擇 [建立部署]。
   b. 針對 Revision type (修訂版類型)，選擇 My application is stored in Amazon S3 (我的應用程式存放在 Amazon S3)。
   c. 對於「修訂」位置，請選擇`s3://my-revision-bucket/httpd_app.zip`。
   d. 對於「修訂」檔案類型，請選擇`.zip`。
   e. 選擇 Create deployment (建立部署)。

   由於 Auto Scaling 群組中現在有一個`InService`執行個體，因此此部署應該可以運作，而且您應該不會再看到錯誤部署失敗，因為找不到您的部署群組的執行個體。

8. 部署成功後，如果您先前將 Auto Scaling 群組擴展至原始容量：

   a. 前往網址 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台，然後從導覽窗格中選擇 Auto Scaling 群組。
   b. 選擇適當的「區域」。
   c. 轉到您的「Auto Scaling」組。
   d. 在群組詳細資料中，選擇編輯。
   e. 將 [所需容量] 設定回其原始值。
   f. 選擇更新。

的錯誤代碼 AWS CodeDeploy

本主題提供有關 CodeDeploy 錯誤的參考資訊。

錯誤程式碼	描述
`AGENT_ISSUE`	部署因為 CodeDeploy 代理程式發生問題而失敗。請確定已在此部署群組的所有執行個體上安裝和執行代理程式。 進一步了解：

錯誤程式碼	描述
	• 確認 CodeDeploy 代理程式正在執行 • 安裝 CodeDeploy 代理程式 • 與 CodeDeploy 代理工作
AUTO_SCALING_IAM_ROLE_PERMISSIONS	與您的部署群組相關聯的服務角色沒有在下列 AWS 服務中執行作業所需的權限。 進一步了解： • 步驟 2：建立服務角色 CodeDeploy • 建立將權限委派給 AWS 服務的角色
HEALTH_CONSTRAINTS	整體部署失敗，因為太多個別執行個體讓部署失敗、太少運作狀態良好的執行個體可用於部署，或部署群組中的一些執行個體發生問題。 進一步了解： • Instance Health • 對執行個體問題進行故障診斷 • 疑難排解 EC2/內部部署問題
HEALTH_CONSTRAINTS_INVALID	部署無法啟動，因為沒有部署組態所定義的運作狀態良好執行個體數目下限。您可以更新部署組態來減少所需的運作狀態良好執行個體數目，或增加此部署群組中的執行個體數目。 進一步了解： • Instance Health • 使用的例證 CodeDeploy

錯誤程式碼	描述
IAM_ROLE_MISSING	部署失敗，因為沒有服務角色具有針對部署群組所指定的服務角色名稱。請確定您使用正確的服務角色名稱。 進一步了解： • 步驟 2：建立服務角色 CodeDeploy • 變更部署群組設定 CodeDeploy
IAM_ROLE_PERMISSIONS	CodeDeploy 沒有擔任角色所需的許可，或者您正在使用的 IAM 角色沒有授予您在 AWS 服務中執行操作的權限。 進一步了解： • 步驟 1：設定 • 步驟 2：建立服務角色 CodeDeploy • 步驟 4：為您的 Amazon EC2 執行個體建立 IAM 執行個體設定檔

錯誤程式碼	描述
NO_INSTANCES	這可能由以下其中一項造成。 • 對於 EC2 /內部部署藍色/綠色部署，如果您使用 Amazon EC2 標籤，則可能未正確設定這些標籤。在您的 CodeDeploy 部署群組中，確定它們包含在藍色執行個體和綠色執行個體中。您可以使用 Amazon EC2 主控台確認執行個體已正確標記。 • 如果您使用 Amazon EC2 Auto Scaling 群組，您的 Auto Scaling 群組可能沒有足夠的容量。請確定您的 Auto Scaling 群組有足夠的容量來進行部署。您可以使用 Amazon EC2 主控台查看其運作狀態良好的執行個體數量，以檢視 Amazon EC2 Auto Scaling 群組的容量。 • 對於 EC2 /內部部署藍色/綠色部署，藍色和綠色叢集的大小可能不相同。請確定兩個機群的大小相同。 進一步了解： • Tagging Instances for Deployments • 教學課程：用 CodeDeploy 於將應用程式部署到 Auto Scaling 群組 • 建立藍色/綠色部署 (主控台) 的應用程式

錯誤程式碼	描述
OVER_MAX_INSTANCES	部署失敗，因為設為部署目標的執行個體數目多於您帳戶允許的執行個體數目。若要減少設為此部署目標的執行個體數目，請更新此部署群組的標籤設定，或刪除一些目標執行個體。或者，您也可以聯絡 AWS Support 要求提高限額。 進一步了解： • 變更部署群組設定 CodeDeploy • CodeDeploy 配額 • 請求提高限制
THROTTLED	部署失敗，因為發出的請求數量超 AWS CodeDeploy 過 IAM 角色允許的數量。請嘗試減少請求數目。 進一步了解： • 查詢 API 請求率
UNABLE_TO_SEND_ASG	部署失敗，因為部署群組未使用其 Amazon EC2 Auto Scaling 群組正確設定。在 CodeDeploy 主控台中，從部署群組中刪除 Amazon EC2 Auto Scaling 群組，然後再次新增該群組。 進一步了解： • 引擎蓋之下： CodeDeploy 和 Auto Scaling 集成

相關主題

疑難排 CodeDeploy

CodeDeploy 資源

下列相關資源可在您使用時協助您 CodeDeploy。

參考指南和支援資源

- AWS CodeDeploy API 參考 — 有關 CodeDeploy 動作和資料類型的說明、語法和使用範例，包括常見參數和錯誤代碼。
- CodeDeploy 技術常見問題解答 — 來自客戶的主要問題 CodeDeploy。
- AWS 支援中心 — 建立和管理 AWS Support 案例的中心。也包含其他資源的連結，例如論壇、技術常見問答集、服務健康狀態和 AWS Trusted Advisor.
- AWS 支援方案 — 方 AWS Support 案相關資訊的主要網頁。
- 聯繫我們 — 有關帳 AWS 單，帳戶，事件，濫用和其他問題的查詢的中心聯繫窗口。
- AWS 網站條款 — 有關我們的版權和商標的詳細資訊；您的帳戶、授權和網站存取權限；以及其他主題。

範例

- CodeDeploy 範例於 GitHub — 的範例與範本案例 CodeDeploy。
- CodeDeploy 詹金斯插件-詹金斯插件。 CodeDeploy
- CodeDeploy agent — 代理程式的開放原始碼版本。 CodeDeploy

部落格

- AWS DevOps 部落格 — 開發人員、系統管理員和架構師的深入解析。

AWS 軟體開發套件和工具

下列 AWS SDK 和工具可透過下列方式支援解決方案開 CodeDeploy發：

- AWS SDK for .NET
- AWS SDK for Java
- AWS SDK for JavaScript

- AWS SDK for PHP
- AWS SDK for Python (Boto)
- AWS SDK for Ruby
- AWS Toolkit for Eclipse — 第 1、2 和第 3 部分。
- AWS Tools for Windows PowerShell— 一組 Windows PowerShell 指令程式，會公開 PowerShell環境 AWS SDK for .NET 中的功能。
- CodeDeploy 中的指令程式 AWS Tools for PowerShell — 一組 Windows PowerShell 指令程式，會公開環境 CodeDeploy 中的 PowerShell 功能。
- AWS Command Line Interface— 用於存取 AWS 服務的統一命令列語法。 AWS CLI 使用單一設定程序來啟用所有支援服務的存取權。
- AWS 開發人員工具 — 開發人員工具和資源的連結，這些工具和資源提供文件、程式碼範例、版本說明和其他資訊，可協助您使用 CodeDeploy 和 AWS建置創新應用程式

文件歷史紀錄

下表說明自上一版《使用者指南》以來，為了支援新功能和增強功能而對本CodeDeploy 使用者指南所做的主要變更。

- 應用程式介面版本:

變更	描述	日期
添加了替代文本（替代文本）	本指南中的所有影像都已更新為包含替代文字。螢幕閱讀程式會閱讀替代文字，讓盲人使用者更容易存取我們的文件。	2024年5月22 日
CodeDeploy 代理程式版本	AWS CodeDeploy 代理程式已更新至版本 1.7.0。如需詳細資訊，請參閱代理程式的版本歷 CodeDeploy 程記錄。	2024年3月6日
變更的指令	不再建議使用這些`sudo service codedeploy-agent status|start|stop`命令，因為它們不用`systemd`於 CodeDeploy 代理程式程序管理，這是最佳作法。若要確`systemd`保使用，請使用`systemctl` 指令，如下列範例所示：`sytemctl start codedeploy-agent`. 下列主題已使用命`systemctl` 令更新：為 Amazon Linux 或 RHEL 安裝 CodeDeploy 代理程式、安裝 Ubuntu 伺服器的 CodeDeploy 代理程式、疑難排解所有已跳過的生命週期事件錯誤，	2024 年 1 月 12 日

	以及在意外刪除時建立新的 CodeDeploy 記錄檔。	
新增主題	在您的生命週期事件指令碼主題中新增了管理 CodeDeploy 代理程式程序和參照檔案。	2024 年 1 月 12 日
CodeDeploy 現在支持區域配置	更新使用區域規劃資訊 CodeDeploy主題建立部署規劃。	2023 年 12 月 7 日
CodeDeploy 現在支援終止部署	已新增在 Auto Scaling 擴充事件期間啟用終止部署主題，以說明終止部署功能。同時更新 EC2 /內部部署的「掛鉤 AppSpec 」區段、為就地部署建立部署群組 (主控台) 建立部署群組，以及針對 EC2 /內部部署藍/綠部署(主控台) 主題建立部署群組以說明此功能。	2023 年 12 月 7 日
固定的 JSON 格式	已修正「資源AppSpec」區段 (僅限 Amazon ECS 和 AWS Lambda 部署) 主題中 JSON 程式碼範例的格式。	2023年12月3日
新增疑難排解主題	添加了 Amazon ECS 部署問題故障排除主題。	2023 年 10 月 24 日
更新了 AppSpec 文件名	已更新CodeDeploy AppSpec 檔案參考，指出 AppSpec 檔案必須`appspec.yml` 為 EC2 /內部部署命名。	2023 年 10 月 5 日

CodeDeploy 現在支援多個負載平衡器	更新了為就地部署建立部署 (主控台) 的部署群組、為 EC2 /內部部署藍/綠部署(主控台) 建立部署群組，以及在 CodeDeploy Amazon EC2 部署的 Elastic Load Balancing 中設定負載平衡器主題以表示支援多個負載平衡器。	2023 年 9 月 26 日
更新了 VPC 主題中的區域	已更新 CodeDeploy與 Amazon Virtual Private Cloud 搭配使用主題中的表格，以顯示其他區域支援。具體而言，亞太區域 (海德拉巴)、亞太區域 (墨爾本)、歐洲 (米蘭)、歐洲 (西班牙) 和歐洲 (蘇黎世) 區域已更新，以顯示對代理程式端點的支援。	2023 年 9 月 22 日
更新資源套件主題中的區域	將下列區 AWS 域新增至「依地區分類的資源套件值區名稱」區段：亞太區域 (大阪)、亞太區域 (海德拉巴)、加拿大 (中部)、歐洲 (西班牙)、歐洲 (蘇黎世)、中東 (阿拉伯聯合大公國)。同時更新了這些區域及其他任何遺失的 IAM 政策。	2023 年 9 月 22 日
簡化代理程式安裝和更新主題	縮短「在 Windows 伺服器上安裝 CodeDeploy 代理程式」並在 Windows 伺服器上更新 CodeDeploy 代理程式主題。移除多餘的 Amazon S3 儲存貯體 URL 和 Amazon S3 複製命令。	2023 年 9 月 21 日

新增亞太區域 (雅加達) 區域	依區域將亞太區域 (雅加達) 新增至資源套件值區名稱。	2023 年 9 月 21 日
CodeDeploy 更新了現有的 AWS 受管策略	受 AWSCodeDeployRole 管理的策略已更新。如需詳細資訊，請參閱 AWS 受管政策的 AWS 更新項目。	2023 年 8 月 16 日
新增限制	新增限CodeDeploy 制主題的限制。此限制為與部署群組相關聯的警示數目上限。	2023 年 8 月 15 日
修正與負載平衡器相關的步驟	已修正為 EC2 /內部部署藍/綠部署建立部署群組 (主控台) 中的指示。負載平衡器步驟現在已標示為選用。	2023 年 8 月 3 日
Amazon ECS 主題中澄清的措辭	闡明了教學課程：將應用程式部署到 Amazon ECS 中的措辭。現在，措辭表示您正在部署應用程式。之前，措辭表明您正在部署 Amazon ECS 服務。	2023 年 8 月 3 日
CodeDeploy 以色列 (特拉維夫) 地區現已推出	CodeDeploy 以色列（特拉維夫）區域（il-central-1）現已推出。數個主題 (包括內含設定 CodeDeploy 代理程式指示的主題) 已更新，以反映此新區域的可用性。	2023 年 7 月 31 日
主題更新	更新了 EC2/內部部署問題疑難排解主題，其中包含有關使用 Runbook 自動執行疑難排解工作的提示。	2023 年 7 月 7 日

主題更新	更新了 Amazon ECS 部署主題的 AppSpec 「資源」部分，其中包含有關任務定義 ARN 的詳細資訊。	2023 年 7 月 7 日
主題更新	更新了步驟 1：使用疑難排解資訊 AWS CLI 在內部部署執行個體主題上安裝和設定。	2023 年 7 月 7 日
主題更新	更新了跨服務混淆副預防主題，其中包含 Amazon ECS 藍/綠部署的相關資訊。 AWS CloudFormation	2023 年 7 月 6 日
主題更新	跨服務混淆副預防主題已更新為 Amazon ECS 藍/綠部署的相關資訊。 AWS CloudFormation	2023 年 7 月 6 日
主題更新	更新 EC2 /內部部署計算平台的預先定義部署組態主題。已新增有關使用 Auto Scaling 群組`CodeDeployDefault.HalfAtATime` 預先定義部署組態行為的備註。	2023 年 6 月 29 日
主題更新	更新 AWS CodeDeploy主題中的基礎結構安全性，以指出傳輸層安全性 (TLS) 通訊協定的新版本下限和建議版本。	2023 年 6 月 28 日
限制更新	已變更下列限制：「EC2 /內部部署現地部署可執行的最大小時間」。如需詳細資訊，請參閱限制	2023 年 6 月 27 日
主題更新	步驟 3：限制 CodeDeploy用戶的權限主題已更新詳細說明。	2023 年 5 月 31 日

CodeDeploy 更新了現有的 AWS 受管策略	受 AWSCodeDeployFullAccess 管理的策略已更新。如需詳細資訊，請參閱 AWS 受管政策的AWS 更新項目。	2023 年 5 月 16 日
CodeDeploy 代理程式版本	AWS CodeDeploy 代理程式已更新至版本 1.6.0。如需詳細資訊，請參閱代理程式的版本歷 CodeDeploy 程記錄。	2023 年 3 月 30 日
CodeDeploy 代理程式版本	AWS CodeDeploy 代理程式已更新至版本 1.5.0。如需詳細資訊，請參閱代理程式的版本歷 CodeDeploy 程記錄。	2023 年 3 月 3 日
Amazon ECS 運算平台更新	亞太區域 (雅加達) 區域現在支援 Amazon ECS 運算平台上的部署。	2023 年 2 月 8 日
CodeDeploy 更新了現有的 AWS 受管策略	受 AWSCodeDeployRole 管理的策略已更新。如需詳細資訊，請參閱 AWS 受管政策的 AWS 更新項目。	2023 年 2 月 3 日
主題更新	「CodeDeploy 搭配 Amazon Virtual Private Cloud 使用」主題已更新為新的和變更的 AWS 區域。	2023 年 2 月 2 日
主題更新	CodeDeploy 現已於亞太區域 (墨爾本) 區域 (ap-southeast-4) 推出。數個主題 (包括內含設定 CodeDeploy 代理程式指示的主題) 已更新，以反映這些新區域的可用性。	2023 年 1 月 26 日

安全性最佳做法更新	[入門使用 CodeDeploy] 區段和其他幾個區段已更新，以符合 AWS 安全性最佳實務。	2023 年 1 月 23 日
CodeDeploy 代理程式版本	AWS CodeDeploy 代理程式已更新至版本 1.4.1。如需詳細資訊，請參閱代理程式的版本歷 CodeDeploy 程記錄。	2022 年 12 月 6 日
新增疑難排解主題	已新增主題，說明如何疑難排解 Windows CodeDeploy 代理程式使用的長檔案路徑所造成的錯誤。如需詳細資訊，請參閱長檔案路徑導致「無此類檔案或目錄」錯誤。	2022 年 12 月 6 日
變更限制	已變更下列限制：「與 AWS 帳戶相關聯的自訂部署組態數目上限」。現在的限制是 200。如需有關限制的詳細資訊，請參閱限制主題。	2022 年 9 月 7 日
CodeDeploy 代理程式版本	AWS CodeDeploy 代理程式已更新至版本 1.4.0。如需詳細資訊，請參閱代理程式的版本歷 CodeDeploy 程記錄。	2022 年 8 月 31 日

修正了一些限制。	已修正下列限制：「與 AWS 帳戶相關聯的同時部署數目上限」現在為 1000。「單一部署中的執行個體數目上限」現在是 1000。'正在進行中且與一個帳戶相關聯的並行部署可以使用的最大執行個體數目' 現在是 1000 個。'與 AWS 帳戶相關聯的自訂部署組態數目上限' 現在是 100。如需有關限制的詳細資訊，請參閱限制主題。	2022 年 8 月 9 日
已新增一個表格，其中顯示每個區域支援的 CodeDeploy 端點。	如需詳細資訊，請參閱搭 CodeDeploy配 Amazon Virtual Private Cloud 使用。	2022 年 4 月 20 日
為 Amazon ECS 藍/綠部署添加了新的限制。	Amazon ECS 藍/綠部署期間，修訂部署到流量轉移到替換環境之間的最大小時數目前為 120 小時。如需詳細資訊，請參閱「限制」主題中的部署。	2022 年 4 月 12 日
添加了一個關於如何防止混淆副問題的話題	如需詳細資訊，請參閱中AWS Identity and Access Management 的 AWS CodeDeploy。	2022 年 3 月 14 日
CodeDeploy 更新了現有的 AWS 受管策略	`AmazonEC2RoleforAWSCodeDeployLimited` 角色已更新。如需詳細資訊，請參閱AWS 受管理的原則更新。	2021 年 11 月 22 日
CodeDeploy 更新了現有的 AWS 受管策略	`AWS CodeDeployRole` 已更新。如需詳細資訊，請參閱AWS 受管理的原則更新。	2021 年 5 月 18 日

CodeDeploy 代理程式版本 1.3.2 版本	代 AWS CodeDeploy 理程式已更新至 1.3.2 版。如需詳細資訊，請參閱代理程式的版本歷 CodeDeploy 程記錄。	2021 年 5 月 6 日
CodeDeploy 支援更新過時的 Amazon EC2 執行個體	CodeDeploy 現在支援自動更新過時的 Amazon EC2 執行個體。如需詳細資訊，請參閱設定部署群組的進階選項。	2021 年 2 月 23 日
CodeDeploy 代理程式版本	AWS CodeDeploy 代理程式已更新至版本 1.3.1。如需詳細資訊，請參閱代理程式的版本歷 CodeDeploy 程記錄。	2020 年 12 月 22 日
CodeDeploy 代理程式版本	AWS CodeDeploy 代理程式已更新至版本 1.3.0。如需詳細資訊，請參閱代理程式的版本歷 CodeDeploy 程記錄。	2020 年 11 月 10 日
CodeDeploy 代理程式版本	AWS CodeDeploy 代理程式已更新至版本 1.2.1。如需詳細資訊，請參閱代理程式的版本歷 CodeDeploy 程記錄。	2020 年 9 月 23 日
CodeDeploy 支援提供支援的 Amazon VPC 端點 AWS PrivateLink	如果您使用 Amazon Virtual Private Cloud (Amazon VPC) 託管 AWS 資源，則可以在 VPC 和. CodeDeploy 您可以使用此連線 CodeDeploy 來啟用與 VPC 上的資源進行通訊，而無需透過公用網際網路。如需詳細資訊，請參閱搭 CodeDeploy 配 Amazon Virtual Private Cloud 使用。	2020 年 8 月 11 日

更新的 CodeDeploy 服務限制	將每個帳戶的應用程式數目和每個應用程式的部署群組數目上限更新為 1000 個。如需 CodeDeploy 服務限制的詳細資訊，請參閱CodeDeploy 限制。	2020 年 8 月 6 日
CodeDeploy 代理程式版本	代 AWS CodeDeploy 理程式已更新至 1.1.2 版。如需詳細資訊，請參閱代理程式的版本歷CodeDeploy 程記錄。	2020 年 8 月 4 日
CodeDeploy 代理 1.1.0 發布和與 Amazon EC2 Systems Manager 集成	現已推出 CodeDeploy 代理程式 1.1.0 版，如需詳細資訊，請參閱 CodeDeploy 代理程式的版本歷程記錄。您現在可以使用 Amazon EC2 系統管理員在 Amazon EC2 或現場部署執行個體上自動管理 CodeDeploy 代理程式安裝和更新。如需詳細資訊，請參閱使用 Amazon EC2 系統管理員安裝 CodeDeploy代理程式。	2020 年 6 月 30 日
CodeDeploy 支援管理Amazon ECS 藍色/綠色部署 AWS CloudFormation	您現在可以透 AWS CloudFormation 過管理 Amazon ECS 藍色/綠色部署。CodeDeploy您可以透過定義綠色和藍色資源，並指定要在 AWS CloudFormation中使用的流量路由和穩定設定來產生部署。如需詳細資訊，請參閱透過建立 Amazon ECS 藍/綠部署。 AWS CloudFormation	2020 年 5 月 19 日

CodeDeploy 支援Amazon ECS 藍/綠部署的加權流量移動	CodeDeploy 現在支援 Amazon ECS 藍/綠部署的加權流量轉移。您可以選擇或建立部署組態，以指定部署中的流量轉移間隔的數目，以及每個間隔中要轉移的流量百分比。下列主題已更新，以反映此變更：Amazon ECS 運算平台上的部署組態。	2020 年 2 月 6 日
更新的安全性、驗證和存取控制主題	的安全性、驗證和存取控制資訊 CodeDeploy 已整理成新的「安全性」一章。如需詳細資訊，請參閱安全性。	2019 年 11 月 26 日
CodeDeploy 支援通知規則	您現在可以使用通知規則，向使用者通知部署中的重要變更。如需詳細資訊，請參閱建立通知規則。	2019 年 11 月 5 日
更新的主題	CodeDeploy 現已於亞太區域 (香港) 區域 (ap-east-1) 推出。數個主題 (包括內含設定 CodeDeploy 代理程式指示的主題) 已更新，以反映此新區域的可用性。您必須明確啟用此區域的存取。如需詳細資訊，請參閱管理 AWS區域。	2019 年 4 月 25 日

更新的主題	AWS CodeDeploy 現在支援 Amazon ECS 服務中容器化應用程式的藍/綠部署。使用新 Amazon ECS 運算平台的 CodeDeploy 應用程式會將容器化應用程式部署到同一 Amazon ECS 服務中的新替代任務集。已新增和更新數個主題以反映此變更，包括AWS CodeDeploy 運算平台概觀、Amazon ECS 運算平台上的部署、Amazon ECS 部署的AppSpec 檔案結構，以及為 Amazon ECS 服務部署建立應用程式 (主控台)。	2018 年 11 月 27 日
更新的 CodeDeploy 代理	AWS CodeDeploy 代理程式已更新至 1.1.1597 版本。如需詳細資訊，請參閱代理程式的版本歷 CodeDeploy 程記錄。	2018 年 11 月 15 日
已更新主控台	本指南中的程序已更新，以符合 CodeDeploy主控台的新設計。	2018 年 10 月 30 日
新的最低支援版本的 CodeDeploy 代理程式	AWS CodeDeploy 代理程式的最低支援版本現在為 1.7.x。如需詳細資訊，請參閱代理程式的版本歷 CodeDeploy 程記錄。	2018 年 8 月 7 日

舊版更新

下表描述 2018 年 6 月前，每個 AWS CodeDeploy 使用者指南版本的重要變更。

變更	描述	變更日期
主題更新	CodeDeploy 現已在歐洲 (巴黎) 區域 (eu-west-3) 區域推出。數個主題 (包括內含設定 CodeDeploy 代理程式指示的主題) 已更新，以反映此新區域的可用性。	2017 年 12 月 19 日
更新主題	CodeDeploy 現已在中國（寧夏）地區推出。 若要在中國 (北京) 區域或中國 (寧夏) 區域使用服務，您必須擁有這些地區的帳戶和憑證。其他 AWS 地區的帳戶和憑證不適用於北京和寧夏地區，反之亦然。 有關中國區域某些資源的資訊，例如 CodeDeploy Resource Kit 值區名稱和 CodeDeploy 代理程式安裝程序，不包含在此版本的CodeDeploy 使用者指南中。 如需詳細資訊： • CodeDeployin 中國 (北京) 地區開始使用 AWS • CodeDeploy 中國地區用戶指南 (英文版本 | 中文版)	2017 年 12 月 11 日
更新主題	CodeDeploy 現在支援部署 Lambda 函數。 AWS Lambda 部署可將傳入流量從現有的 Lambda 函數轉移到更新的 Lambda 函數版本。您可以選擇或建立部署組態，以指定部署中的流量轉移間隔數目，以及每個間隔中要移動的流量百分比。 AWS Lambda AWS 無伺服器應用程式模型 (AWS SAM) 支援部署，因此您可以使用 S AWS AM 部署偏好設定來管理部署期間流量轉移的方式。 AWS Lambda 新增及更新數個主題以反映此變更，包括 CodeDeploy 運算平台概觀、在 AWS Lambda 運算平台上進行部署、建立 AWS Lambda 運算平台部署 (主控台)、 建立 AWS Lambda 函數部署的應用程式 (主控台)和為 AWS Lambda 部署新增 AppSpec 檔案。	2017 年 11 月 28 日
新主題	CodeDeploy 現在支援直接部署到已安裝 CodeDeploy 代理程式的本機電腦或執行個體。您可以在本機測試部署，如果有錯誤，請使用 CodeDeploy 代理程式錯誤記錄檔對其進行除錯。您也可以使用本端部署來測試應用程式修訂版本的完	2017 年 11 月 16 日

變更	描述	變更日期
	整性、 AppSpec 檔案內容等。如需詳細資訊，請參閱 使用 CodeDeploy 代理程式驗證本機電腦上的部署套件。	
更新主題	CodeDeploy 部署群組中 Elastic Load Balancing 器的支援已擴充，包括適用於藍/綠部署和就地部署的網路負載平衡器。您現在可以為部署群組選擇 Application Load Balancer 載平衡器、Classic Load Balancer 或 Network Load Balancer。負載平衡器是藍色/綠色部署的必要項目。若為就地部署則為選擇性項目。更新數個主題以反映此支援，包括Integrating CodeDeploy with Elastic Load Balancing、建立就地部署的應用程式 (主控台)、部署先決條、Integrating CodeDeploy with Elastic Load Balancing和建立就地部署的應用程式 (主控台)。	2017 年 9 月 12 日
更新主題	CodeDeploy 部署群組中 Elastic Load Balancing 器的支援已擴充，包括適用於藍/綠部署和就地部署的應用程式負載平衡器。您現在可以為部署群組選擇 Application Load Balancer 載平衡器和 Classic Load Balancer。負載平衡器是藍色/綠色部署的必要項目。若為就地部署則為選擇性項目。數個主題 (包括 Integrating CodeDeploy with Elastic Load Balancing、建立應用程式 CodeDeploy和建立部署群組 CodeDeploy) 皆已更新，以反映此新增支援。	2017 年 8 月 10 日
新增與更新主題	CodeDeploy 現在支援使用多個標記群組來識別要包含在部署群組中的執行個體聯集和交集。若您使用單一標籤群組，任何可透過群組中至少一個標籤識別的執行個體都會包含在部署群組中。若您使用多個標籤群組，只有可透過每個群組中至少一個標籤識別的執行個體才會包含在部署群組中。如需將執行個體新增至部署群組的新方法相關資訊，請參閱Tagging Instances for Deployments。其他更新以反映此支援的主題包括建立就地部署的應用程式 (主控台)、建立藍色/綠色部署 (主控台) 的應用程式、建立就地部署的部署群組 (主控台)、為EC2/內部部署藍/綠部署建立部署群組 (主控台)、Deployments和 教學課程：用 CodeDeploy 來部署應用程式 GitHub中的 步驟 5：建立應用程式和部署群組。	2017 年 7 月 31 日

變更	描述	變更日期
更新主題	在 Windows 伺服器執行個體上安裝 CodeDeploy 代理程式的另外兩種方法已新增至安裝 Windows 伺服器的 CodeDeploy 代理程式。除了 Windows 命 PowerShell 令之外，現在還可以使用直接 HTTPS 連結和使用 Amazon S3 複製命令來下載安裝檔案的說明。在檔案下載或複製到執行個體之後，您可以手動執行安裝。	2017 年 7 月 12 日
更新主題	CodeDeploy 改善了管理 GitHub 帳戶和儲存庫連線的方式。您現在可以建立並儲存最多 25 個與 GitHub帳戶的連線，以便將 CodeDeploy 應用程式與 GitHub 儲存庫建立關聯。每個連線可支援多個儲存庫。您可以建立最多 25 個不同 GitHub 帳戶的連線，或是建立一個以上的單一帳戶連線。將應用程式連線到 GitHub帳戶後，即可 CodeDeploy 管理所需的存取權限，而不需要您採取任何進一步的動作。更新指定儲存在存 GitHub 放庫中之修訂版本的相關資訊、CodeDeploy 與整合 GitHub及 教學課程：用 CodeDeploy 來部署應用程式 GitHub，以反映此支援。	2017 年 5 月 30 日
更新主題	在過去，如果 CodeDeploy 代理程式偵測到目標位置中的檔案不屬於最近一次成功部署的應用程式修訂版本的一部分，依預設會失敗目前的部署。 CodeDeploy 現在提供代理程式如何處理這些檔案的選項：部署失敗、保留內容或覆寫內容。 使用建立部署 CodeDeploy已更新以反映此支援，且新區段復原現有內容的行為已新增至使用以下方式重新部署和復原部署 CodeDeploy。	2017 年 5 月 16 日

變更	描述	變更日期
更新主題	Elastic Load Balancing 中的 Classic Load Balancer 現在可以使用 CodeDeploy 主控台或指派給部署群組 AWS CLI。在就地部署期間，負載平衡器會防止網際網路流量路由至即將部署到的目標執行個體，並會在部署到該執行個體完成後讓執行個體再次開始接受流量。數個主題已更新以反映此支援，包括與其他 AWS 服務整合、Integrating CodeDeploy with Elastic Load Balancing、建立就地部署的應用程式 (主控台)、建立就地部署的部署群組 (主控台)和AppSpec「掛鉤」部分。新的章節疑難排解失敗 ApplicationStop BeforeBlockTraffic、或 AfterBlockTraffic 部署生命週期事件已新增至故障診斷指南。	2017 年 4 月 27 日
更新主題	Elastic Load Balancing 中的 Classic Load Balancer 現在可以使用 CodeDeploy 主控台或指派給部署群組 AWS CLI。在就地部署期間，負載平衡器會防止網際網路流量路由至即將部署到的目標執行個體，並會在部署到該執行個體完成後讓執行個體再次開始接受流量。數個主題已更新以反映此支援，包括與其他 AWS 服務整合、Integrating CodeDeploy with Elastic Load Balancing、建立就地部署的應用程式 (主控台)、建立就地部署的部署群組 (主控台)和AppSpec「掛鉤」部分。新的章節疑難排解失敗 ApplicationStop BeforeBlockTraffic、或 AfterBlockTraffic 部署生命週期事件已新增至故障診斷指南。	2017 年 5 月 1 日

變更	描述	變更日期
更新主題	CodeDeploy 現已在中國（北京）地區提供。 若要在中國 (北京) 區域或中國 (寧夏) 區域使用服務，您必須擁有這些地區的帳戶和憑證。其他 AWS 地區的帳戶和憑證不適用於北京和寧夏地區，反之亦然。 有關中國區域某些資源的資訊，例如 CodeDeploy Resource Kit 值區名稱和 CodeDeploy 代理程式安裝程序，不包含在此版本的CodeDeploy 使用者指南中。 如需詳細資訊： • CodeDeployin 中國 (北京) 地區開始使用 AWS • CodeDeploy 中國地區用戶指南 (英文版本 | 中文版)	2017 年 3 月 29 日
新增與更新主題	已引入數個新主題，以反映藍/綠部署的新 CodeDeploy 支援，部署群組 (原始環境) 中的執行個體會被不同的執行個體集 (取代環境) 取代的部署方法。 藍/綠部署概述提供所CodeDeploy使用藍色/綠色方法的高階說明。其他新主題包括建立藍色/綠色部署 (主控台) 的應用程式、為EC2/內部部署藍/綠部署建立部署群組 (主控台)，以及在 Elastic Load Balancing 中為 CodeDeploy Amazon EC2 部署設定負載平衡器。 數個主題也已更新，包括使用建立部署 CodeDeploy、使用中的部署組態 CodeDeploy、建立應用程式 CodeDeploy、使用中的部署群組 CodeDeploy、使用中的部署 CodeDeploy和AppSpec 「掛鉤」部分。	2017 年 1 月 25 日
新增與更新主題	新主題說明如何使用透過產生的定期重新整理臨時認證來驗證和註冊內部部署執行個體 AWS Security Token Service。使用命 register-on-premises-instance 令 (IAM 工作階段 ARN) 註冊內部部署執行個體此方法可針對大型現場部署執行個體群，提供比在每個執行個體上僅使用靜態 IAM 使用者的登入資料更佳的支援。Working with On-Premises Instances也已更新，以反映這項新支援。	2016 年 12 月 28 日

變更	描述	變更日期
更新主題	CodeDeploy 現已在歐洲 (倫敦) 區域 (eu-west-2) 推出。數個主題 (包括內含設定 CodeDeploy 代理程式指示的主題) 已更新，以反映此新區域的可用性。	2016 年 12 月 13 日
更新主題	CodeDeploy 現已在加拿大 (中部) 區域 (ca-central-1) 推出。數個主題 (包括內含設定 CodeDeploy 代理程式指示的主題) 已更新，以反映此新區域的可用性。	2016 年 12 月 8 日
更新主題	CodeDeploy 現已在美國東部 (俄亥俄) 區域 (us-east-2) 推出。數個主題 (包括內含設定 CodeDeploy 代理程式指示的主題) 已更新，以反映此新區域的可用性。	2016 年 10 月 17 日
新增主題	身份驗證和存取控制新章節提供有關使用 AWS Identity and Access Management (IAM) 的全面資訊， CodeDeploy 以及透過使用登入資料來協助保護資源的存取安全。這些登入資料提供存取 AWS 資源所需的許可，例如從 Amazon S3 儲存貯體擷取應用程式修訂版，以及讀取 Amazon EC2 執行個體上的標籤。	2016 年 10 月 11 日
更新主題	更新 Windows 伺服器上的 CodeDeploy 代理程式已更新，以反映 Windows 伺服器的新 CodeDeploy 代理程式更新程式的可用性。在 Windows 伺服器執行個體上安裝時，更新程式會定期檢查是否有新版本。當偵測到新版本時，更新程式會移除目前的代理程式版本 (若有安裝的話)，再安裝最新版本。	2016 年 10 月 4 日

變更	描述	變更日期
更新主題	CodeDeploy 現在與 Amazon CloudWatch 警示整合，如果指定的警示狀態發生變更，連續數個期間 (如警示臨界值中所指定)，就可以停止部署。 CodeDeploy 現在也支援在符合特定條件 (例如部署失敗或啟動的警示) 時自動回復部署。 更新數個主題以反映這些變更，包括建立應用程式 CodeDeploy、建立部署群組 CodeDeploy、變更部署群組設定 CodeDeploy、Deployments, 使用以下方式重新部署和復原部署 CodeDeploy和產品與服務整合 CodeDeploy，並另外加入新的主題使用 CloudWatch 警示監控部署 CodeDeploy。	2016 年 9 月 15 日
新增與更新主題	CodeDeploy 現在提供與 Amazon CloudWatch 活動的集成。現在，當在部署狀態或屬於部署群組的執行個體狀態下偵測到變更時，您現在可以使用 E CloudWatch vents 啟動一或多個動作。 CodeDeploy 您可以合併叫用 AWS Lambda 函數的動作、發佈到 Kinesis 串流或 Amazon SNS 主題、將訊息推送至 Amazon SQS 佇列的動作；或者接著觸發 CloudWatch 警示動作。如需詳細資訊，請參閱 使用 Amazon CloudWatch 事件監控部署。	2016 年 9 月 9 日
主題更新	主題Integrating CodeDeploy with Elastic Load Balancing和與其他 AWS 服務整合已更新，以反映額外的負載平衡選項。 CodeDeploy 現在支援 Elastic Load Balancing 中提供的 Classic Load Balancer 和應用程式負載平衡器。	2016 年 8 月 11 日
主題更新	CodeDeploy 現已於亞太區域 (孟買) 區域 (位於 ap-south-1) 推出。數個主題 (包括內含設定 CodeDeploy 代理程式指示的主題) 已更新，以反映此新區域的可用性。	2016 年 6 月 27 日

變更	描述	變更日期
主題更新	CodeDeploy 現已於亞太區域 (首爾) 區域 (ap-northeast-2) 推出。數個主題 (包括內含設定 CodeDeploy 代理程式指示的主題) 已更新，以反映此新區域的可用性。 重新組織目錄，以包括執行個體、部署組態、應用程式、部署群組、修訂，以及部署的各節。已為 CodeDeploy自學課程加入新區段。為增加可用性，已將數個較長的主題 (包括CodeDeploy AppSpec 檔案參考和疑難排 CodeDeploy) 分割成數個較短的主題。 CodeDeploy代理程式的組態資訊已移至新主題CodeDeploy 用戶端組態參考。	2016 年 6 月 15 日
新增與更新主題	的錯誤代碼 AWS CodeDeploy提供 CodeDeploy 部署失敗時可能顯示的某些錯誤訊息的相關資訊。 下列疑難排 CodeDeploy中的各節已更新，可更進一步輔助解決部署問題： • Amazon EC2 Auto Scaling 群組中的 EC2 執行個體無法啟動並收到錯誤「活動訊號逾時」 • 避免將多個部署群組與單一 Amazon EC2 Auto Scaling 群組建立關聯	2016 年 4 月 20 日
主題更新	CodeDeploy 現已在南美洲 (聖保羅) 區域 (sa-east-1) 推出。數個主題 (包括內含設定 CodeDeploy代理程式指示的主題) 已更新，以反映此新區域的可用性。 與 CodeDeploy 代理工作已更新以反映新的:max_revision: 組態選項，您可以使用此選項來指定您希望代理程式封存之部署群組的應用程式修訂數目。 CodeDeploy	2016 年 3 月 10 日

變更	描述	變更日期
新增與更新主題	CodeDeploy 現在支援將觸發程序新增至部署群組，以接收與該部署群組中部署或執行個體相關事件的通知。這些通知會傳送給訂閱 Amazon 簡單通知服務主題的收件者，您已參與觸發器動作的一部分。您也可以使用在您自己的自訂通知工作流程內觸動觸發時建立的 JSON 資料。如需詳細資訊，請參閱 Monitoring Deployments with Amazon SNS Event Notifications。 程序已更新，反映重新設計的應用程式詳細資訊頁面。 疑難排 CodeDeploy中的如果執行個體在部署期間終止，在最多一小時內部署不會失敗一節已更新。 CodeDeploy 配額已更新以反映可與單一應用程式產生關聯之部署群組數目的修訂限制、運作狀態最小執行個體設定的允許值，以及 AWS SDK for Ruby.	2016 年 2 月 17 日
新增與更新主題	CodeDeploy 現在可在美國西部 (加利佛尼亞北部) 區域 (us-west-1) 使用。數個主題 (包括包含設定 CodeDeploy 代理程式指示的主題) 已更新，以反映此新區域的新增情況。 選擇 CodeDeploy 存放庫類型列出並說明目前支援的存放庫類型 CodeDeploy。這個新主題也會在引進其他支援的儲存庫類型時更新。 管理 CodeDeploy 代理程式作已更新為新增至執行個體的新`.version`檔案相關資訊，以報告代理程式的目前版本，以及 CodeDeploy 代理程式支援版本的相關資訊。 程式碼範例的語法反白功能 (包括 JSON 和 YAML 範例) 已新增至使用者指南。 將應用程式規格檔案新增至修訂 CodeDeploy已重新組織為 step-by-step 指示。	2016 年 1 月 20 日

變更	描述	變更日期
新主題	在不同的 AWS 帳戶中部署應用程式會說明啟動屬於您其他組織帳戶部署的安裝必要條件和程序，而無需該帳戶的完整登入資料。這對針對不同用途 (例如其中一個與開發和測試環境有關，另一個則和生產環境有關) 使用多個帳戶的組織非常有用。	2015 年 12 月 30 日
主題更新	產品與服務整合 CodeDeploy主題已經過重新設計。它現在包含了社群整合範例的區段，其中包含與 CodeDeploy整合相關的部落格文章和影片範例清單。	2015 年 12 月 16 日
主題更新	CodeDeploy 現已於亞太區域 (新加坡) 區域 (ap-south east-1) 推出。數個主題 (包括內含設定 CodeDeploy 代理程式指示的主題) 已更新，以反映此新區域的可用性。	2015 年 12 月 9 日
主題更新	與 CodeDeploy 代理工作已更新，以反映 CodeDeploy代理程式組態檔中的新`:proxy_uri:` 選項。 CodeDeploy AppSpec 檔案參考已更新，包括使用新環境變數 (`DEPLOYMENT_GROUP_ID`) 的相關資訊。勾點指令碼可在部署生命週期事件期間存取該環境變數。	2015 年 12 月 1 日
主題更新	步驟 2：建立服務角色 CodeDeploy已更新，以反映針對建立服務角色的新程序，以 CodeDeploy 及納入其他改良功能。	2015 年 11 月 13 日
主題更新	CodeDeploy 現已在歐洲 (法蘭克福) 區域 (eu-central-1) 推出。數個主題 (包括內含設定 CodeDeploy 代理程式指示的主題) 已更新，以反映此新區域的可用性。 疑難排 CodeDeploy主題已更新，包含確認執行個體上時間設定準確的相關資訊。	2015 年 10 月 19 日
新增主題	AWS CloudFormation CodeDeploy供參考的範本發布以反映對 CodeDeploy 行動的新 AWS CloudFormation 支持。 建立 Primary Components主題並引進「目標修訂」的定義。	2015 年 10 月 1 日

變更	描述	變更日期
主題更新	建立部署群組 CodeDeploy已更新，反映使用萬用字元搜尋尋找部署群組執行個體的能力。 Instance Health已更新，釐清「最低良好運作狀態執行個體」的概念。	2015 年 8 月 31 日
主題更新	CodeDeploy 現已於亞太區域 (東京) 區域 (ap-northeast-1) 推出。數個主題 (包括內含設定 CodeDeploy 代理程式指示的主題) 已更新，以反映此新區域的可用性。	2015 年 8 月 19 日
主題更新	CodeDeploy 現在支援部署到 RHEL (RHEL) 現場部署執行個體和 Amazon EC2 執行個體。如需詳細資訊，請參閱下列主題： • CodeDeploy 代理程式支援的作業系統 • 使用的例證 CodeDeploy • 教學課程：部署 WordPress 至 Amazon EC2 執行個體 (Amazon Linux 或紅帽企業 Linux、macOS 或 Unix) • 教學課程：使用 CodeDeploy (Windows 伺服器、Ubuntu 伺服器或 RHEL) 將應用程式部署到內部部署執行個體	2015 年 6 月 23 日
主題更新	CodeDeploy 現在提供部署指令碼可在部署期間使用的一組環境變數。這些環境變數包括目前 CodeDeploy 應用程式名稱、部署群組和部署生命週期事件等資訊，以及目前的 CodeDeploy 部署識別碼。如需詳細資訊，請參閱 CodeDeploy AppSpec 檔案參考中 AppSpec 「掛鉤」部分一節的結尾。	2015 年 5 月 29 日

變更	描述	變更日期
主題更新	CodeDeploy 現在在 IAM 中提供一組 AWS 受管政策，您可以使用這些政策，而不必自行手動建立對等政策。其中包含： • 一種原則，可讓使用者 CodeDeploy 僅註冊修訂版本，然後透過部署修訂 CodeDeploy。 • 提供使用者完整 CodeDeploy資源存取權的原則。 • 為使用者提供資 CodeDeploy源唯讀存取權的原則。 • 附加至服務角色的政策， CodeDeploy 以便透過 Amazon EC2 標籤、現場部署執行個體標籤或 Amazon EC2 Auto Scaling 群組名稱來識別 Amazon EC2 執行個體，並相應地將應用程式修訂部署到這些執行個體。 如需詳細資訊，請參閱身分驗證與存取控制中的客戶受管政策範例一節。	2015 年 5 月 29 日
主題更新	CodeDeploy 現已在歐洲 (愛爾蘭) 區域 (eu-west-1) 和亞太區域 (雪梨) 區域 (ap-southeast-2) 推出。數個主題 (包括內含設定 CodeDeploy 代理程式指示的主題) 已更新，以反映這些新區域的可用性。	2015 年 5 月 7 日
新增主題	CodeDeploy 現在支援部署到現場部署執行個體和 Amazon EC2 執行個體。新增下列主題，描述這項新支援： • Working with On-Premises Instances • 教學課程：使用 CodeDeploy (Windows 伺服器、Ubuntu 伺服器或 RHEL) 將應用程式部署到內部部署執行個體 • Working with On-Premises Instances	2015 年 4 月 2 日
新主題	新增CodeDeploy 資源。	2015 年 4 月 2 日

變更	描述	變更日期
主題更新	更新疑難排 CodeDeploy： • 新章節長時間執行的程序可能導致部署失敗會說明您可以採取的步驟，用以識別和處理因長時間執行程序而造成的部署失敗。 • 本一般的 Amazon EC2 Auto Scaling 故障排除節 CodeDeploy 已更新，顯示 CodeDeploy 代理程式的 Amazon EC2 Auto Scaling 逾時邏輯已從五分鐘增加到一小時。 • 新不匹配的 Amazon EC2 自 Auto Scaling 生命週期掛鉤可能會導致對 Amazon EC2 自動 Auto Scaling 群組的自動部署停止或失敗章節說明您可以採取的步驟，以識別和解決失敗的自動部署到 Amazon EC2 Auto Scaling 群組。	2015 年 4 月 2 日
主題更新	下列主題已更新，以反映建立自訂政策，然後將其附加至 IAM 中的使用者和角色的新建議： • 設定要使用的 Amazon EC2 執行個體 CodeDeploy • 步驟 4：為您的 Amazon EC2 執行個體建立 IAM 執行個體設定檔 • 步驟 2：建立服務角色 CodeDeploy 新增兩個章節至疑難排 CodeDeploy： • 一般故障診斷檢查清單 • 視窗 PowerShell 指令碼預設無法使用 64 位元版本 PowerShell 的視窗 CodeDeploy AppSpec 檔案參考中的 AppSpec 「掛鉤」部分一節已更新，可更精確的說明可用的部署生命週期事件。	2015 年 2 月 12 日

變更	描述	變更日期
主題更新	新增章節至 疑難排 CodeDeploy：Amazon EC2 Auto Scaling 群組中的 EC2 執行個體無法啟動並收到錯誤「活動訊號逾時」。 已將 CloudBees 區段加入至產品與服務整合 CodeDeploy。	2015 年 1 月 28 日
主題更新	疑難排 CodeDeploy已新增下列各節： • 使用某些文字編輯器建立 AppSpec檔案和 shell 指令碼可能會導致部署失敗 • 使用 macOS 的 Finder 套用應用程式修訂可能導致失敗 • 疑難排解失敗 ApplicationStop BeforeBlockTraffic、或 AfterBlockTraffic 部署生命週期事件 • 疑難排解失敗的 DownloadBundle 部署生命週期事件 UnknownError：未開啟以供讀取 • 一般的 Amazon EC2 Auto Scaling 故障排除	2015 年 1 月 20 日
新增主題	產品與服務整合 CodeDeploy一節已更新，其中包含下列主題： • CodeDeploy 與 Amazon EC2 Auto Scaling 集成 • 教學課程：用 CodeDeploy 於將應用程式部署到 Auto Scaling 群組 • Monitoring Deployments • Integrating CodeDeploy with Elastic Load Balancing • CodeDeploy 與整合 GitHub • 教學課程：用 CodeDeploy 來部署應用程式 GitHub	2015 年 1 月 9 日

變更	描述	變更日期
主題更新	• 新增 CodePipeline使用自動部署 CodeDeploy一節至 CodeDeploy 與整合 GitHub。您現在可以在儲 GitHub 存庫中的原始程式碼發生變更時自動觸發部署。 • 新增 疑難排解 Amazon EC2 Auto Scaling 問題一節至疑難排 CodeDeploy。本新章節說明如何解決部署到 Amazon EC2 Auto Scaling 群組時的常見問題。 • 新的子章節 "files Examples" (files 範例) 已新增至 CodeDeploy AppSpec 檔案參考中的 AppSpec 「檔案」區段 (僅適用於 EC2 /內部部署) 一節。這個新小節包含數個範例，說明如何在部署期間使用 AppSpec 檔案`files`區段指示 CodeDeploy 將特定檔案或資料夾複製到 Amazon EC2 執行個體上的特定位置。	2015 年 1 月 8 日
新主題	Monitoring Deployments已新增。 CodeDeploy 與這項服務整合 AWS CloudTrail，可擷取您帳戶 CodeDeploy 中由或代表您 AWS 帳戶發出的 API 呼叫，並將日誌檔案傳送到您指定的 Amazon S3 儲存貯體。	2014 年 12 月 17 日
初始公有版本	這是《CodeDeploy 使用者指南》的首次公開發行。	2014 年 11 月 12 日

AWS 詞彙表

有關最新 AWS 術語，請參閱AWS 詞彙表 參考文獻中的AWS 詞彙表。

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。